Optimization of P2 meshes and applications

Rémi Feuillet, Adrien Loseille, Frédéric Alauzet

Optimization of P^2 meshes and applications

Rémi Feuillet[a], Adrien Loseille[a], Frédéric Alauzet[a,*]

[a]Gamma *Project, INRIA Saclay-Île-de-France, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France*

Abstract

Mesh optimization techniques are a way to locally modify the mesh in order to improve it with respect to a given quality criterion. To this end, this work presents the generalization of two mesh quality-based optimization operators to P^2 meshes. The generalized operators consist in mesh smoothing and generalized swapping. With the use of these operators, P^2 mesh generation starting from a P^1 mesh is more robust and P^2 connectivity-change moving mesh methods for large displacements are now possible.

Keywords: High-order meshes, Mesh optimization, Connectivity-change moving mesh methods

1. Introduction

In numerical simulation, unstructured meshes are commonly used. More specifically, in Computational Fluid Dynamics (CFD) they are used to solve real-world problems common in industrial and governmental institutions. In the last decade high-order resolution methods (*e.g.* continuous Galerkin [1], discontinuous Galerkin [2], spectral differences [3]) have been used. To preserve the order of convergence of these methods, a high-order discrete representation of the geometry is required [4]. These meshes are curved to best represent (and to align with) the boundary of the geometric domain. In this context, the generation and the processing of high-order meshes is necessary.

To generate high-order meshes, several approaches exist. The first approach was tackled twenty years ago [5] for both surface and volume meshing. At this moment the main idea was to curve the entire mesh. The same problem was revisited a few years later in [6] for bio-medical applications. In these first approaches and in those that follow, the underlying idea is to use a P^1 mesh and elevate it to the desired order. Some make use of a PDE or variational approach to do so [7, 8, 9]. Others are based on optimization and smoothing operations and start from a P^1 mesh with a constrained P^k curved boundary. They then try to generate a suitable P^k mesh [10, 11, 12]. In all these techniques, the main concept is to deform an initial P^1 mesh to create a high-order one. However, the validity of the high-order mesh is critical. Until the work presented in [13, 14], no real

*Corresponding author

Email addresses: remi.feuillet@inria.fr (Rémi Feuillet), adrien.loseille@inria.fr (Adrien Loseille), frederic.alauzet@inria.fr (Frédéric Alauzet)

approach was proposed to strongly deal with the validity of high-order elements. The novelty of these approaches was to view the geometrical elements and their Jacobian as Bézier entities. Based on the properties of the Bézier representation, the validity of the element can be determined in a strong sense, while the other methods were only using a sampling of the Jacobian to determine its sign.

A connectivity-change moving-mesh method proposed in [15] relies on both mesh deformation and mesh optimization techniques. This approach has been proven to be very efficient. In this work, the motion of vertices is first computed thanks to a linear elasticity model and then the position of these vertices is changed via local mesh optimization operators such as generalized swapping and mesh smoothing. To apply this connectivity-change moving-mesh method to high-order meshes, these two operators need to be generalized to high-order meshes.

In this paper, P^2 mesh quality-based optimization operators are presented. They are a generalization of the P^1 operators and rely on the resolution of a local optimization problem. These two operators can be applied to improve P^2 mesh generation starting from a P^1 and enable high-order connectivity-change moving mesh methods.

The paper is outlined as follows. Section 2 defines what is a high-order mesh and sets up validity and quality criteria. Section 3 treats P^2 mesh optimization. Section 4 shows two applications with examples of P^2 mesh optimization. The first one is an improvement of a P^2 mesh generation technique that curves a P^1 mesh thanks to a high-order linear elasticity solver and the second one describes a P^2 connectivity-change moving mesh method also using a high-order linear elasticity solver. Finally, Section 4 discusses conclusions and perspectives driven by this work.

2. High-order element, validity and quality criteria

To properly define P^2 quality-based optimization operators, it is fundamental to properly define quality and validity criteria for high-order elements.

2.1. Definition of a high-order element

In general, a finite element is defined (see [1]) by the triplet $\{K, \Sigma_K, V_h\}$ where K denotes the geometric element (triangle, etc), Σ_K the list of nodes of K, and V_h, the space of the *shape functions*, here it will be the Lagrange polynomial functions (or interpolants). To properly define the geometry and these functions, a reference space (that can also be seen as a parameter space) is defined where all coordinates are between 0 and 1. In this space, the reference element is denoted $\widehat{K}$ and has a fixed (and sometimes uniform) distribution of nodes. The element K, also called physical element, is thus the image of $\widehat{K}$ via a mapping, denoted F_K (see Figures 1 and 3). More specifically, for each point M of K, there is a point $\widehat{M}$ of $\widehat{K}$ such that $M = F_K(\widehat{M})$. The exact expression is defined by:

$$M = \sum_{i=1}^{n} \phi_i(\widehat{M}) A_i,$$

where n is the number of nodes, $A_i = F_K(\widehat{A}_i)$ with $\widehat{A}_i$ the nodes of the reference element which map to A_i the nodes of the physical element, and ϕ_i are the Lagrange polynomial

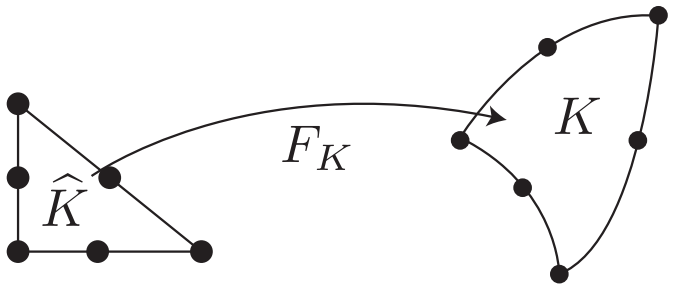

Figure 1: Mapping F_K from $\widehat{K}$ to K for a P^2 triangle.

functions such that:

$$\phi_i(\widehat{A}_j) = \delta_{ij}, \quad \forall i, \quad \text{and} \quad \sum_{i=1}^{n} \phi_i = 1.$$

Let us have a look at the case of the d-simplices (*e.g.* triangles when $d = 2$ and tetrahedra when $d = 3$) that are the elements that will be used in this work.

Case of the triangle. The triangle is defined thanks to a two-dimensional triangular reference element. To define a complete finite element of degree k on a triangle the number of (distinct) nodes needs to be equal to $n = \frac{(k+1)(k+2)}{2}$. In this case, the reference coordinates $(\widehat{x}, \widehat{y})$ can be used to define the triangle barycentric coordinates (u, v, w) with the formula: $u = 1 - \widehat{x} - \widehat{y}$, $v = \widehat{x}$ and $w = \widehat{y}$.
A point M of the triangle can also be expressed in Bézier form [16] using the Bernstein polynomials B^k_{ijl} as:

$$M = \sum_{i+j+l=k} B^k_{ijl}(u, v, w) P_{ijl},$$

with $B^k_{ijl}(u, v, w) = \frac{k!}{i!j!l!} u^i v^j w^l$. The points $(P_{ijl})_{i+j+l=k}$, also noted $(C_i)_{1 \le i \le n}$ (see Figure 2) are the Bézier control points and are directly related to the nodes $(A_i)_{1 \le i \le n}$.
The computation of these coefficients is done by solving the following linear system:

$$\sum_{i+j+l=k} B^k_{ijl}(u_\alpha, v_\alpha, w_\alpha) P_{ijl} = A_\alpha, \ \forall \alpha \in \{1, .., n\},$$

where $(u_\alpha, v_\alpha, w_\alpha)$ are the barycentric coordinates corresponding to A_α. The solution of this linear system can be written as a matrix vector product:

$$C = M^{2D}_{B2L} A.$$

where C is a $2n$ vector containing all the coordinates of the control points, A is a $2n$ vector containing all the coordinates of the Lagrange points and M^{2D}_{B2L} a $2n \times 2n$ matrix. Note that in the case of a uniform distribution of the nodes on the reference element, M^{2D}_{B2L} is a sparse matrix as a consequence of the properties of the Bernstein polynomials (see [17]).
For instance, to compute C_4 (P_{110}) in Figure 2, we obtain the formula:

$$C_4 = \frac{4A_4 - A_1 - A_2}{3^{\,2}}.$$

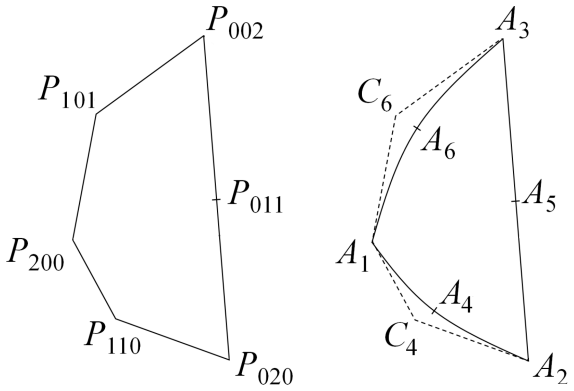

Figure 2: Correspondence between the control points and the nodes for a P^2 triangle.

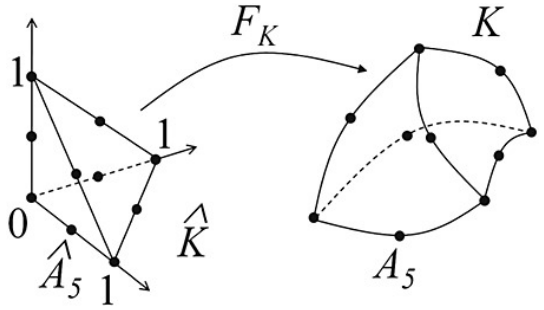

Figure 3: Mapping F_K from $\widehat{K}$ to K for a P^2 tetrahedron.

Case of the tetrahedron. The tetrahedron is defined thanks to a three-dimensional tetrahedron reference element. To define a complete finite element of degree k on a tetrahedron the number of (distinct) nodes needs to be equal to $n = \frac{(k+1)(k+2)(k+3)}{6}$. In this case, the reference coordinates $(\widehat{x}, \widehat{y}, \widehat{z})$ can be used to define the tetrahedron barycentric coordinates (u, v, w, t) with the formula: $u = 1 - \widehat{x} - \widehat{y} - \widehat{z}$, $v = \widehat{x}$, $w = \widehat{y}$, $t = \widehat{z}$.

Similar to the case of triangles, a point M of the tetrahedron can also be expressed in Bézier form using the Bernstein polynomials B^k_{ijlm} as:

$$M = \sum_{i+j+l+m=k} B^k_{ijlm}(u, v, w, t) P_{ijlm},$$

with $B^k_{ijlm}(u, v, w, t) = \frac{k!}{i!j!l!m!} u^i v^j w^l t^m$.

Again, the computation of the control points is done by solving the following linear system:

$$\sum_{i+j+l+m=k} B^k_{ijlm}(u_\alpha, v_\alpha, w_\alpha, t_\alpha) P_{ijlm} = A_\alpha, \ \forall \alpha \in \{1, .., n\},$$

where $(u_\alpha, v_\alpha, w_\alpha, t_\alpha)$ are the barycentric coordinates corresponding to A_α. The solution of this linear system can be written as a matrix vector product:

$$C = M^{3D}_{B2L} A.$$

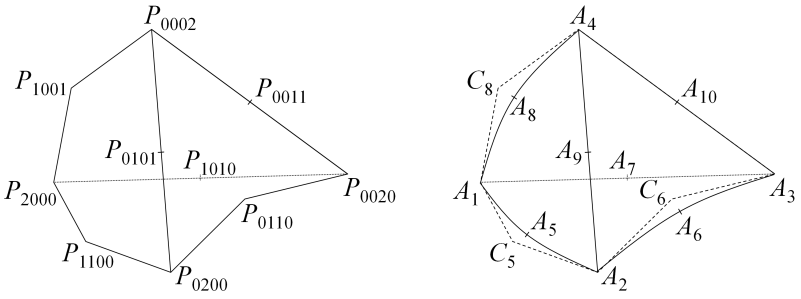

Figure 4: Correspondence between the control points and the nodes for a P^2 tetrahedron.

with the same conventions as in the case of the triangle, except that the size is $3n$. Note that because of the properties of the Bernstein polynomials and for a uniform distribution of the nodes in the reference element, M^{3D}_{B2L} is a sparse matrix (see [17]).
For instance, to compute C_5 (P_{1100}) in Figure 4, we obtain the formula:

$$C_5 = \frac{4A_5 - A_1 - A_2}{2}.$$

In general and in the following sections, $(A_i)_{1\leq i\leq d+1}$ are called vertices, $(A_i)_{d+2\leq i\leq n}$ are called nodes, and $(C_i)_{1\leq i\leq n}$ are called control points.

2.2. Validity of a high-order simplicial element

The validity of an element means that the associated mapping F_K is a diffeomorphism. It can be ensured if the minimum of the determinant $\mathcal{J}_K$ of the Jacobian matrix of the mapping F_K is strictly positive everywhere inside the element [13]. In the case of a simplicial element, Jacobian $\mathcal{J}_K$ can be written as polynomial of degree $d \times (k-1)$ in the barycentric coordinates of the simplex, where d is the dimension of the simplex and k the degree of the simplex. When the element is of degree 1 (e.g. straight-sided), it simply means that the oriented volume/area is strictly positive. The Jacobian can also be expressed in the Bernstein polynomial basis (Theorem of Panzoult, see [13]).

Case of the triangle. In the case of a P^k triangle, the Jacobian[1] can be expressed as follows:

$$\begin{aligned}\mathcal{J}^{2D}_K(u,v,w) = \det\left(\frac{\partial F_K}{\partial \widehat{x}}, \frac{\partial F_K}{\partial \widehat{y}}\right) &= \det\left(\frac{\partial F_K}{\partial v} - \frac{\partial F_K}{\partial u}, \frac{\partial F_K}{\partial w} - \frac{\partial F_K}{\partial u}\right) \\ &= \det\left(\frac{\partial F_K}{\partial v} - \frac{\partial F_K}{\partial w}, \frac{\partial F_K}{\partial w} - \frac{\partial F_K}{\partial u}\right),\end{aligned}$$

[1]It can be noted that F_K is dependent on the choice of the reference element and that consequently three transformations are possible. However, the Jacobian of these three expressions is the same which is why we choose arbitrary one transformation for our analysis. This observation holds for the tetrahedron.

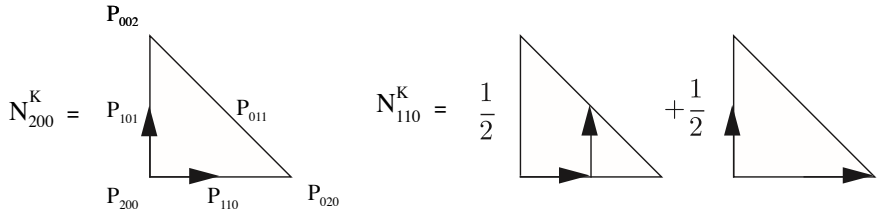

Figure 5: Vectors involved in the determinant for the computation of a corner (left) and an edge (right) control coefficients of a P^2 triangle.

Now, as we have:

$$\frac{\partial F_K}{\partial u} = k \sum_{i+j+l=k-1} B_{ijl}^{k-1}(u,v,w) P_{(i+1)jl},$$

and the same formula for the other variables. The Jacobian can be rewritten as:

$$\begin{aligned}
&\mathcal{J}_K^{2D}(u,v,w) \\
&= k^2 \det\left(\sum_{i+j+l=k-1} B_{ijl}^{k-1}(u,v,w) \overrightarrow{P_{ij(l+1)}P_{i(j+1)l}}, \sum_{i+j+l=k-1} B_{ijl}^{k-1}(u,v,w) \overrightarrow{P_{(i+1)jl}P_{ij(l+1)}} \right) \\
&= k^2 \det\left(\sum_{i+j+l=k-1} B_{ijl}^{k-1}(u,v,w) \overrightarrow{P_{(i+1)jl}P_{i(j+1)l}}, \sum_{i+j+l=k-1} B_{ijl}^{k-1}(u,v,w) \overrightarrow{P_{(i+1)jl}P_{ij(l+1)}} \right).
\end{aligned}$$

Finally, by applying the multiplicative properties of the Bernstein polynomials, we arrive at:

$$\mathcal{J}_K^{2D}(u,v,w) = \sum_{i+j+l=2(k-1)} B_{ijl}^{2(k-1)}(u,v,w) N_{ijl}^K,$$

where:

$$N_{ijl}^K = k^2 \sum_{\substack{i_1+i_2=i \\ j_1+j_2=j \\ l_1+l_2=l}} \frac{i!j!l!((k-1)!)^2}{(2(k-1))!i_1!i_2!j_1!j_2!l_1!l_2!} \det(\overrightarrow{P_{ijl}P_{(i-1)(j+1)l}}, \overrightarrow{P_{ijl}P_{(i-1)j(l+1)}}).$$

N_{ijl}^K will be called a control coefficient and their expression as determinants give them a geometrical meaning. For P^2 (see Figure 5), a corner and an edge control coefficients [17] are for example:

$$N_{200}^K = 4 \det(\overrightarrow{P_{200}P_{110}}, \overrightarrow{P_{200}P_{101}}), \tag{1}$$

$$N_{110}^K = 2 \det(\overrightarrow{P_{200}P_{110}}, \overrightarrow{P_{110}P_{011}}) + 2 \det(\overrightarrow{P_{110}P_{020}}, \overrightarrow{P_{200}P_{101}}). \tag{2}$$

These control coefficients also appear in the computation of the area of a P^k-triangle.

Indeed, we have the following result on a d-simplex K^d[1]:

$$\int_{K^d} \prod_{i=1}^{d+1} {u_i}^{\alpha_i} \mathrm{d}K = \frac{\prod_{i=1}^{d+1} \alpha_i!}{(d + \sum_{i=1}^{d+1} \alpha_i)!} d! |K^d|, \tag{3}$$

where $\alpha_i \in \mathbb{N}$ and $(u_i)_{1 \leq i \leq d+1}$ denote its barycentric coordinates. Now, if we apply the formula on the reference triangle, we have:

$$\int_{\widehat{K}} u^i v^j w^l \mathrm{d}K = \frac{i!j!l!}{(2+i+j+l)!}.$$

Consequently, the area of the triangle is given by:

$$\begin{aligned} |K| &= \int_{\widehat{K}} \mathcal{J}_K^{2D}(u,v,w)\mathrm{d}K = \sum_{i+j+l=2(k-1)} N_{ijl}^K \int_{\widehat{K}} B_{ijl}^{2(k-1)}(u,v,w)\mathrm{d}K \\ &= \sum_{i+j+l=2(k-1)} N_{ijl}^K \frac{(2(k-1))!}{i!j!l!} \frac{i!j!l!}{(2+i+j+l)!}, \end{aligned}$$

and finally:

$$|K| = \frac{2}{(2(k-1)+1)(2(k-1)+2)} \sum_{i+j+l=2(k-1)} \frac{N_{ijl}^K}{2}. \tag{4}$$

In other words, the area of a P^k-triangle is the average of all the control coefficients multiplied by the area of the reference element that is $\frac{1}{2}$.
For the P^k-tetrahedron, we have also:

$$\mathcal{J}_K^{3D}(u,v,w,t) = \sum_{i+j+l+m=3(k-1)} B_{ijlm}^{3(k-1)}(u,v,w,t) N_{ijlm}^K,$$

where N_{ijlm}^K are the control coefficient of the Jacobian. These coefficients can also be explicitly found and have the same kind of geometrical meaning (see [17] for more details).

Case of the tetrahedron. In a same manner, using Eq. (3), we have that the volume of the tetrahedron is:

$$|K| = \frac{6}{(3(k-1)+1)(3(k-1)+2)(3(k-1)+3)} \sum_{i+j+l+m=3(k-1)} \frac{N_{ijlm}^K}{6}, \tag{5}$$

e.g. it is the average of all the control coefficients multiplied by the area of the reference element that is $\frac{1}{6}$.

65

Validity check procedure. The Bernstein polynomials are always positive on the reference element and realize a partition of unity. Consequently, a sufficient condition to prove that $\mathcal{J}_K$ is strictly positive everywhere is to ensure that all N_{ijlm}^K are strictly positive, but this condition is too strong. On the contrary, if a N_{ijlm}^K is negative in a corner, it means that $\mathcal{J}_K$ is negative somewhere inside the element as N_{ijlm}^K is the exact value of

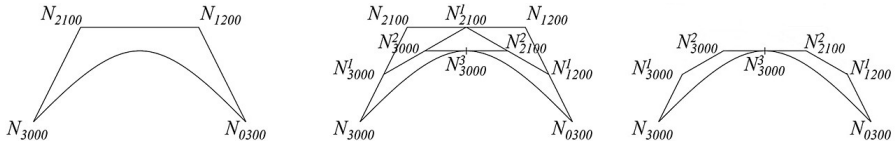

Figure 6: De Castljau's refinement on two control coefficients of the Jacobian curve on a curved edge of a P^2 tetrahedron. Left, the initial curve with its control coefficients. Middle, construction of the new control coefficients. Right, the initial curve divided in two curves with their control coefficients.

the Jacobian in this corner [13]. However, if a control coefficient lying on an edge or on a face or in a volume is negative, it does not mean that $\mathcal{J}_K$ is negative somewhere inside the element. We cannot conclude on the positiveness of the Jacobian without any further analysis. In this case, a few iterations of a subdivision algorithm [13, 14] are required to have more accurate bounds of the Jacobian. The idea of this refinement on an edge/face/volume is to create new control coefficients from the initial ones. These control coefficients can define several curves whose union is the Jacobian curve on the edge/face/volume. This way, more accurate bounds can be found.

For instance, let us consider an edge on a P^2 tetrahedron (see Figure 6, left) with a negative edge control coefficient on it. The chosen algorithm in this case is the De Casteljau's algorithm [18]. This algorithm can be decomposed into three steps (see Figure 6, middle, K superscripts are omitted for clarity):

$$N^1_{3000} = \frac{N_{3000} + N_{2100}}{2} \quad N^1_{2100} = \frac{N_{2100} + N_{1200}}{2} \quad N^1_{1200} = \frac{N_{1200} + N_{0300}}{2},$$

$$N^2_{3000} = \frac{N^1_{3000} + N^1_{2100}}{2} \quad N^2_{2100} = \frac{N^2_{2100} + N^2_{1200}}{2},$$

$$N^3_{3000} = \frac{N^2_{3000} + N^2_{2100}}{2}.$$

By construction, $N^3_{3000} = \mathcal{J}_K(\frac{1}{2}, \frac{1}{2}, 0, 0)$. So, if $N^3_{3000} \leq 0$ then the element is invalid. Otherwise, $N_{3000}, N^1_{3000}, N^2_{3000}, N^3_{3000}$ (resp. $N^3_{3000}, N^2_{2100}, N^1_{1200}, N_{0300}$) defines a P^3 curve representing the Jacobian between $\mathcal{J}_K(1, 0, 0, 0)$ and $\mathcal{J}_K(\frac{1}{2}, \frac{1}{2}, 0, 0)$ (resp. $\mathcal{J}_K(\frac{1}{2}, \frac{1}{2}, 0, 0)$ and $\mathcal{J}_K(0, 1, 0, 0)$) (see Figure 6, right). Consequently, this analysis can be done again on these two sub-curves. If central control coefficients are positive, the Jacobian on the sub-element is valid. Otherwise, it is not possible to conclude, as in the initial case. The process is therefore reapplied to this sub-curve. If one sub-curve gives an invalid Jacobian then the whole curve is invalid.

This way, a recursive method to find the sign of the minimum of the Jacobian on the edge is established. Note that the example was done for a Jacobian of degree 3 but works for any degree.

In the case of negativity inside a face/volume, the De Casteljau's algorithm can also be applied. However, the proposed subdivision is not suitable for all the cases. Indeed, in the case of triangular faces, the algorithm provides a conforming subdivision in three of the faces, but this subdivision does not subdivide on the edges of the face. This is an issue as we want a subdivision that subdivides on the edges of the triangle as well. In this case, the high-order conforming subdivision is explicitly computed to perform the analysis.

From a more general point of view, the validation of high-order finite-element meshes for complex three-dimensional cases is really expensive. Indeed, Table 1 shows the number of determinants that need to be checked to deal with the validity of a high-order tetrahedron. Whereas the cost of a high-order mesh from a solver point of view would be to sample the value of the Jacobian at the Gauss (integration) points (*e.g.* the number of nodes) of the geometry, we realize that the effective number of coefficients to be checked for the validity of such meshes is way higher. Indeed, we need to make sure that the mesh is valid everywhere no matter what the order is, considering that the order of the solver and the position of the integration nodes are not known *a priori*. In particular this shows that the generation of valid tetrahedral meshes for degrees higher than three is really expensive. Notably, the ratio between the number of determinants to check and the number of nodes drastically increases when the degree increases.

Tetrahedron	P^1	P^2	P^3	P^4	P^5
# nodes	4	10	20	35	56
Degree of the Jacobian	1	3	6	9	12
# control coef.	1	20	84	220	455
# determinants check	1	64	1 000	**8 000**	**42 875**
# dets/(# nodes)	-	6.4	50	**228.6**	**765.6**

Table 1: Statistics about the validity check of a high-order tetrahedron.

2.3. A quality measure for high-order simplicial elements

Once the validity of the element is known, it is interesting to consider a quality criterion for the shape of the element. The chosen one is the one proposed in [19]:

$$Q_{P^2} = \underbrace{\alpha \frac{hS_k}{V_k}}_{①} \underbrace{\frac{\max(V_1, V_k)}{\min(V_1, V_k)}}_{②} \underbrace{\left(\frac{N^K_{max}}{N^K_{min}}\right)^{1/d}}_{③} \in [1, \infty),$$

with:

- d the dimension, S_k the exterior surface of the polyhedron (in 2D, the half perimeter of the polygon) defined by nodes and vertices $(A_i)_{1 \leq i \leq n}$, *i.e.* the surface of the linear decomposition (made using control points) of the high-order element.
- V_k the volume (resp. surface) of the high-order simplex defined previously. It is computed using Formulas (4) and (5).
- h the element's largest edge P^k-length (e.g the length of the union of straight-sided lines defined by the nodes),
- V_1 the volume/surface of the equivalent P^1 element, e.g the element defined by the vertices.
- N^K_{min} (resp. N^K_{max}) the smallest (resp. largest) control coefficient of the Jacobian of the element.

- α is a normalization factor, dependent of the dimension such that $Q_{P^2} = 1$ for a regular simplex, $\alpha = \frac{\sqrt{3}}{6}$ in 2D and $\alpha = \frac{\sqrt{3}}{36}$ in 3D.

This quality function is actually a product of 3 terms. The derivation of each of the terms is detailed in Appendix, following the strategy of [19]. (1) is only a generalization of the P^1 quality function and measures the gap to the regular element. (2) measures the distance between the volume of the curved element and the volume of the straight element and ensures the function to be greater than 1 [19]. And, finally (3) gives a measure of the distortion of the element, it can detect if the element is invalid or almost invalid by taking an infinite value. Note that if the element is straight, the standard P^1 quality function [20] is recovered:

$$Q_{P^2} = Q_{P^1} = \alpha \frac{hS_1}{V_1} = \alpha \frac{h}{\rho} \in [1, \infty),$$

where ρ is the inradius of the straight element. Also, this quality function can easily be extended to anisotropic meshes. Based on these definitions, this element-wise quality measure is between 1 and infinity[2]. The closer the element quality is to 1, the better the quality.

3. Optimization of P^2 meshes

In the same way as we want to have an optimal P^1 mesh in terms of quality, we want to have an optimal P^k mesh. Several optimization techniques exist to correct an invalid P^k mesh [12, 10] and to optimize the geometrical accuracy [21].
The idea here is to extend two classic mesh quality-based optimization operators [15] to P^2 meshes to improve its quality.

3.1. P^2 swap operator

The swap operator (see Figure 7) locally changes the connectivity of the mesh in order to improve its quality.
In 2D, it consists in flipping an edge shared by two triangles to form two new triangles with the same four vertices (see Figure 7 left).
In 3D, two types of swap exist: face and edge swapping. The face swapping is the extension of the 2D edge swapping, it consists in replacing the common face of two neighboring tetrahedra by the edge linking the opposite vertices to the face of each tetrahedron, also called $2 \rightarrow 3$. The edge swapping is a bit different. First, the shell of the edge to delete (e.g. the set of elements containing this edge) is constructed. From a shell of size n, a non-planar pseudo-polygon formed by n vertices is obtained. The swap consists in deleting the edge, generating a triangulation of the polygon and creating two tetrahedra for each triangle of the triangulation thanks to the two extremities of the former edge. These swaps are designated as $n \rightarrow m$ with $n \geq 3$, where n is the initial number of tetrahedra and $m = 2(n-2)$ is the final number of tetrahedra. Figure 7 shows the case of three tetrahedra around an edge.

[2] Note that in several studies, the quality function is between 0 and 1. However, we prefer to consider values between 1 and infinity (*i.e.* the inverse value, like in [20]) as it uses a greater interval and thus emphasize, from a qualitative point of view, the comparison between two elements of different quality. This fact is illustrated later in this paper with the charts, Figures 13,16.

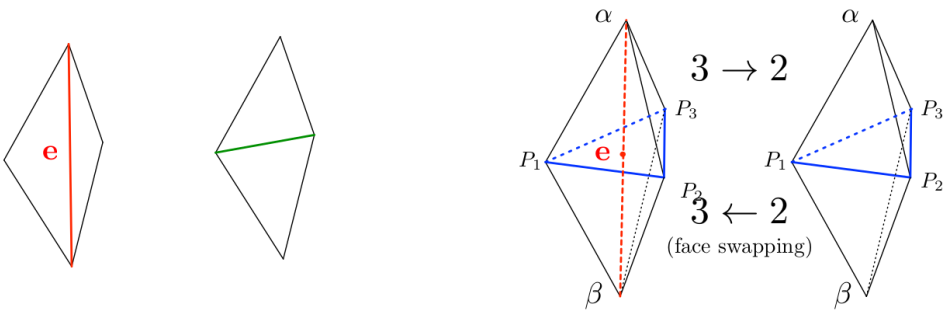

Figure 7: *Left*, the swap operation in 2D. *Right*, edge swap $3 \to 2$ and face swap $2 \to 3$. For all these pictures, shells are in *black*, old edges are in *red*, new edges are in *green*.

For each possible swapped configuration, if the worst quality of all the elements of the shell is improved, the configuration is kept unless another swapped configuration of the shell provides a better quality improvement. In other words, the optimization is only performed if $Q^{new} < cQ^{old}$ where $Q^{new} = \max_{K^{new}} Q(K^{new})$ is the quality of the studied swapped configuration, $Q^{old} = \max_{K^{old}} Q(K^{old})$ is the quality of the initial configuration and c *a quality improvement coefficient*. In general, the coefficient c is set to 0.99 as we are looking for an improvement of the quality of the studied configuration. When it comes to connectivity-change moving-mesh algorithms in 3D, the strategy proposed in [22] is reused and c can be set greater than one. It is usually set between 1.2 and 1.6. This means that we allow swaps to produce a small local degradation of the shell's worst quality. This choice was done in the first place for P^1 cases as it avoids to be stuck in local minima in terms of global quality improvement and thus helps to swap elements farther away. Indeed, if this small degradation is not allowed, then the moving-mesh algorithm may fail. As the P^1 case is included in the P^2 algorithm, it appears natural to keep this feature in the P^2 case.
To generalize the swap to P^2 meshes, the inner nodes of the edges of the shell have to be taken into account. For instance, in 2D, there is one node on the swapped edge and if we want the swap to be performed, we first need to find an optimal position for the node in the swapped configuration and then check if this configuration improves the quality function (see Figure 8). The key feature is therefore to find a functional whose optimum will give the optimal position for the node in the swapped configuration in terms of quality.

In this context, the idea is to find a simple and smooth functional that will be easy to optimize. The quality function is not a good candidate as it is not smooth. We propose to consider the following functional (inspired from the work of [12]):

$$f(\mathbf{X}) = \sum_{K \in \mathcal{S}(e)} \sum_{i+j+l+m=d(k-1)} \omega_{ijlm} \left(\frac{N^K_{ijlm}(\mathbf{X})}{d!\ V_1} - 1 \right)^2, \tag{6}$$

where d is the dimension, k the degree (in the following $k = 2$), $\mathbf{X} = [\mathbf{x_1}, ..., \mathbf{x_n}]$ is a *set* of n coordinates to be optimized, $\mathbf{X} \to N^K_{ijlm}(\mathbf{X})$ is a function that depends, in the worst case on two variables of $\mathbf{X}$, $\mathcal{S}(e)$ is the shell of the initial edge e (e.g. the set of

elements K containing e), ω_{ijlm} are weight factors associated to each control coefficient that measure their importance in guaranteeing the validity of an element (their choice is explained later for each kind of element), and V_1 is the volume of the straight-sided element deduced from K. Note that $\mathbf{X}$ can either be empty (in which case no optimization is required), or contains more than one node's coordinates as it represents the coordinate of the created edges of the shell.
In 2D, $\mathbf{X}$ is always a singleton and is simply noted $\mathbf{x}$, weights ω_{ijl} $(m = 0)$ are set to 2 for the corner coefficients and equal to 1 elsewhere. This choice is made because the negativity of one corner coefficients is equivalent to the invalidity of an element where it is not true for the other coefficients. In this case, f has the following properties : it is a positive definite quadratic form as $\mathbf{x} \to N^K_{ijl}(\mathbf{x})$ is linear in $\mathbf{x}$, which means that the functional has a unique minimum. Also, on every regular swap configuration, the minimum of f in $\mathbf{x}$ is the same as the minimum of the worst quality of the swapped shell in $\mathbf{x}$. Using the result of the optimization problem in the swap configuration gives therefore a very good approximation of the best configuration that can be obtained. Since the best swap configuration is found, we are able to conclude if this swap will improve the quality or not.
In 3D, weights ω_{ijlm} are set to 4 for a corner control coefficient, 2 for an edge control coefficient and 1 otherwise. This choice is made by analogy with the 2D case. Also, in 3D both edge and face control coefficients are at stake. A hierarchy is set between them as it is more important to guarantee positive control coefficient on edges than on faces. In this work, considered swaps are $2 \to 3$, $3 \to 2$ and $4 \to 4$ which means that the functional of the problem is in the worst case quadratic. This is a consequence of the fact that control coefficients have a linear dependence with respect to a given control coefficient. Also, the problem is positive definite on a regular configuration[3]. Note that

[3] By a regular configuration, we mean a configuration where none of the triangles is degenerated.

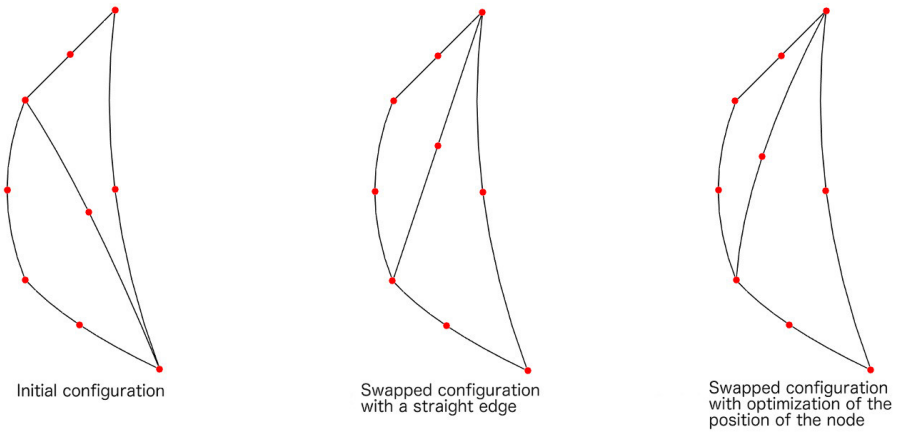

Figure 8: Three steps of P^2 swap in 2D. In the shown case, we have the following qualities $Q^{ini}_{P^2} =$ 10.45 and 6.04 (left), then $Q^{mid}_{P^2} = 5.78$ and 4.48 (middle) and finally $Q^{end}_{P^2} = 4.78$ and 3.72 (right).

these swaps represent $\sim 95\%$ of the swaps performed during a P^1 mesh optimization. For the other swaps ($5 \rightarrow 6$, $6 \rightarrow 8$), the problem begins to be highly costly in term of CPU and is consequently not worth it. Indeed, several factors appear at this point. First, the number of unknowns for the optimization problem increases as the number of high-order nodes to optimize is greater than one. Second, due to the combinatorial number of possible swapped configurations, the number of tests highly increases and so does the number of optimization problem to solve. In the P^1 case, a quick reject procedure can be set up by noticing that a tetrahedron can be present in several configurations. If this tetrahedron provides a poor quality, it rejects automatically all the configurations containing it. This is impossible to do so in P^2 as the optimal position of the inner nodes has no reason to be the same between each of the configurations. The obtained tetrahedra are then unique for each configuration. Therefore, we prefer to perform multiple passes of simple reconnections ($2 \rightarrow 3$, $3 \rightarrow 2$, $4 \rightarrow 4$) that will recover results similar to using more complex reconnections ($5 \rightarrow 6$, $6 \rightarrow 8$).
The resolution of these optimization problems is performed thanks to a L-BFGS algorithm [23]. Like several optimization methods, this method requires the computation of the gradient of f with respect to the coordinates. The computation of the gradient of f mainly relies on the computation of the gradient of N^K_{ijlm}. The computation of this gradient is done in two steps. First, the gradient with respect to the control points is computed. As the control coefficient are determinants based on linear combinations of the control points, it is always possible to rewrite them so that a control point appears at most once in the determinant. In this case, the computation of the gradient is straightforward. Then, by composition of the gradient (the correspondence between Lagrange points and control points is linear), we have:

$$\nabla_{A_n} N^K_{ijlm} = ((M_{L2B})^T \nabla_C N^K_{ijlm})_n$$

where A_n is n^{th} index of the node of coordinates x in K, $\nabla_C N^K_{ijlm}$ is the gradient of N^K_{ijlm} with respect to the control points and M_{L2B} is the matrix which converts Lagrange points into Control points.
Solving this optimization problem gives a solution that tends to minimize the distortion of the Jacobian of the involved elements. However, it does not guarantee a node position that creates a valid configuration. Indeed, like non-convex shells in P^1, there exists shells where no valid solutions can be found, whatever the position of the node is.
Finally, note that even if the quality function is not used for the optimization problem, it is mandatory to keep using it for the decision process so that it degenerates into classic swap operator when the elements are straight.

3.2. P^2 mesh smoothing

Mesh smoothing is a technique that consists in relocating some points inside the mesh with the aim of improving the quality of the elements. In P^1, the idea is to relocate each vertex M_i inside its ball of elements (see Figure 9). For each element K_j in the ball of M_i, noted $\mathcal{B}(M_i)$, the opposite face to M_i denoted by F_j gives an optimal position M^{opt}_j.

Degenerative cases occur when, for instance two consecutive edges (high-order nodes included) of the shell are aligned.

Then, the vertex is relocated considering a weighted average of the proposed positions. If the proposed new location of the vertex does not improve the ball configuration in term of quality, then a relaxation is performed to check if an improved configuration exists between the original location and the new one. The optimal configuration is computed in 3D as follows:

$$M_j^{opt} = G_j + \sqrt{\frac{2}{3}} h_j \frac{\mathbf{n}_j}{||\mathbf{n}_j||}, \tag{7}$$

where G_j is the gravity center of F_j, h_j is the average length of the edges of F_j, and $\mathbf{n}_j$ is the outward normal to F_j. The proposed position is then computed with:

$$M_i^{opt} = \frac{\sum_{K_j \in \mathcal{B}(P_i)} \max(Q_{P^1}(K_j), Q_{\max}) M_j^{opt}}{\sum_{K_j \in \mathcal{B}(M_i)} \max(Q_{P^1}(K_j), Q_{\max})},$$

where $Q_{\max}$ is a parameter to be defined. Here $Q_{\max} = 10$. This way, it avoids really bad quality elements to impact too much the relocation of the considered point. Note that if every computed configuration decreases the quality, the smoothing is not performed.

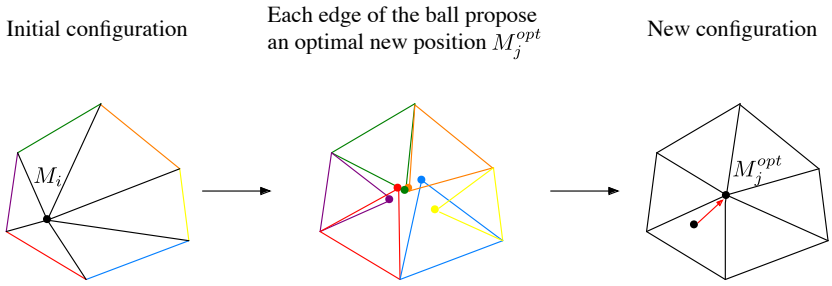

Figure 9: Laplacian smoothing in two dimensions. Each element of the ball of considered vertex M_i suggests an optimal position M_j^{opt}. The resulting new optimal position M_j^{opt} computed as a weighted average of all these proposed locations using Relation (7).

To extend it to P^2 meshes, the edges' node needs to be taken into account. The idea here, is to perform two independent smoothing operations:

- a vertex smoothing,
- a node smoothing.

Vertex smoothing. The vertex smoothing is simply a generalization of the P^1 smoothing. This strategy uses the fact the underlying P^1 mesh quality drives the P^2 mesh quality. The optimal position of the vertex is thus computed in the same way as in P^1 and using P^1 quality weights, and the vertex is located exactly in the same way as before. In order to be consistent with the P^1 vertex smoothing and to preserve straight edges within the ball that are initially straight, the displacement of all the inner nodes of the cavity ball is set to half of the value of the displacement of the central vertex (see Figure 10). In the exact same way as in P^1 if the final configuration does not improve the P^1 quality of the

ball, relaxation is performed between the original location and the new one. Relaxation is also used if the final configuration is not valid in P^2.

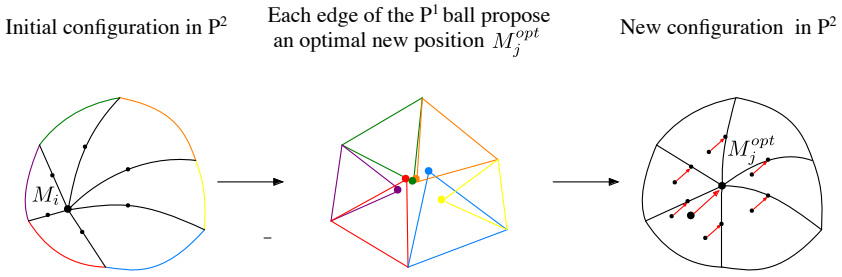

Figure 10: P^2 Laplacian smoothing in two dimensions. Each element of the P^1 ball of considered vertex M_i suggests an optimal position M_j^{opt}. The resulting new optimal position M_j^{opt} is computed as a weighted average of all these proposed locations. The new position of the nodes of the internal edges of the P^2 ball is then deduced by proportionality.

Node smoothing. The optimization of the node position follows the same algorithm as in the P^2 swap operator to find its optimal position. For this purpose, the functional f given by Relation (6) can be re-used to find the optimal node position (see Figure 11). In this case, there is always only one node coordinates to optimize and consequently the optimization problem is quadratic. In the exact same way as in P^1, if the final configuration does not improve the quality, relaxation is performed between the original location and the new one.

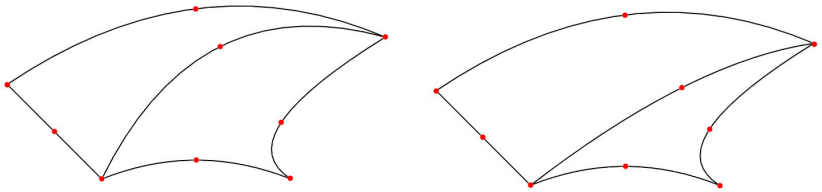

Figure 11: P^2 node smoothing in two dimensions. The optimal position of the node of the central edge is computed solving an optimization problem. Left, the initial configuration, right, the optimized configuration. In the shown case, we have the following initial qualities $Q_{P^2}^{ini} = 11.34$ and 4.69 (left) and the following final qualities $Q_{P^2}^{end} = 8.36$ and 2.65 (right).

3.3. P^2 mesh optimization strategy

Using these two operators, it is then possible to propose a mesh optimization strategy. In this work, we do it in the following order:

1. the vertex smoothing,

2. the node smoothing,
3. the edge swapping.

The choice is done on purpose: the vertex smoothing would give a nicer global behavior of the mesh but may locally negatively impact the quality because it does not really take into account the high-order nodes. The node smoothing is there to improve the quality of the P^2 elements and is also a way to maximize the fact that the initial shell of an edge is optimal before trying any swap on it. If the mesh is straight (or P^1), this part is skipped. Finally, note that swaps perform both edge flipping and node smoothing in the swapped configuration. The latter is indeed automatically handled by the resolution of the optimization problem. Note that this process can be applied several times inside a loop.

4. Applications

4.1. P^2 mesh generation by curving an initial P^1 mesh

Most of the techniques to generate an high-order mesh is to start from a P^1 mesh and then to curve it, in a way or another, in order to obtain a P^k mesh [24, 7, 8, 10]. The main reason to use a post-treatment is that all existing P^1 mesh generation algorithms can be reused. It would be harder to implement a directly high-order mesh generator. Let us think about a boundary recovery procedure [25] with curved entities. To curve meshes, a lot of models are available : PDE or variational models [7, 8, 9], smoothing and/or optimization procedures [10, 11, 12], etc. Our choice here is to use the linear elasticity equation as a model for the motion of the vertices to generate a P^k mesh from a P^1 mesh. For this purpose, let us consider the linear elasticity equation with Dirichlet boundary conditions:

$$\nabla \cdot (\sigma(\mathcal{E})) = 0\,, \quad \text{with} \quad \mathcal{E} = \frac{\nabla\xi + \nabla\xi^T}{2}\,, \tag{8}$$

where σ, and $\mathcal{E}$ are respectively the Cauchy stress and strain tensors, ξ is the Lagrangian displacement. The Cauchy stress tensor follows the Hooke's law for isotropic homogeneous medium.

Here, Dirichlet boundary conditions represent the gap between the P^k-nodes of the initial straight boundary elements and their position on the *real* boundary. For mesh boundary vertices, the gap is equal to 0.

To compute the gap at the nodes, a continuous representation of the surface mesh is required. It can be either provided by CAD/analytical model or deduced from initial P^1 mesh via a cubic reconstruction technique [26]. Once Dirichlet boundary conditions are set, the high-order finite element linear elasticity code is called. The use of a high-order Finite Element (FE) resolution rather than on a subdivided P^1 mesh aims the degrees of freedom to be intrinsically represented. This gives a more physical consistency to the obtained motion.

The elasticity problem using the high-order Finite Element Method (FEM) provides the new position of the internal vertices and nodes. It is then used to generate the high-order mesh by moving the vertices and nodes of the initial straight mesh with the associated values in the elasticity solution vector. If some elements remain invalid after optimization, it is always due to non suitable boundary displacements. In this case, a relaxation

is performed. It consists in finding the invalid element and considering its associated degrees of freedom (boundary included). Their displacement from their initial P^1 position is considered and is reduced of 25%. Then the validity of all the elements containing at least one of these degrees of freedom is checked. If one of them is invalid, then the previous process is again applied to it. This procedure is done until no more elements are invalid and if an element is visited several times, then the factor of relaxation of its degrees of freedom is increased at each visit (50%,75% and finally 100%). The process is summarized by Algorithm 1.

Algorithm 1 Quadratic mesh curving algorithm

1. Generate a P^1 mesh.
2. Perform P^1 mesh optimization pre-processing: generalized swapping and vertex smoothing following the strategy of Section 3.3.
3. Transform the P^1 mesh into a (straight) P^2 mesh by adding the extra degrees of freedom on the middle of the edges of the mesh.
4. Perform cubic reconstruction of the boundary or use its analytical representation to get a boundary displacement vector $\mathbf{d}_{|\partial\Omega}$ and set it as Dirichlet boundary conditions for the linear elasticity equation.
5. Given the boundary conditions, solve linear elasticity equation on the (straight) P^2 mesh with the P^2 FEM and deduce a volume displacement vector $\mathbf{d}_{|\Omega}$.
6. Generate the (curved) P^2 mesh by applying the vector $\mathbf{d}_{|\Omega}$ to all the vertices of the initial straight mesh.
7. Perform P^2 mesh optimization post-processing: generalized swapping and node/vertex smoothing following the strategy of Section 3.3.
8. Check validity of P^2 elements and locally relax the FEM solution if necessary or desired until it is valid as explained in Section 4.1.

The major prerequisite with this method is that the deformed mesh has to be only composed of isotropic or almost isotropic elements. In this context, the use of the elasticity problem is efficient and always provides a valid mesh. Optimization in the pre-processing makes elements more isotropic and therefore helps curvature process to be more robust whereas optimization in the post-processing improves the quality of the mesh and untangle invalid elements if any. Some results in P^2 are presented hereafter.

Straightness criterion. In addition to some qualitative observations, a *straightness criterion* is set up to measure the impact of the mesh curving process on the volume elements. For a given threshold ϵ, the criterion states if an element is straight or not. If we consider, a high-order element K of the mesh, its high-order nodes are denoted $(A_i)_{d+2\leq i\leq n}$. A straight-sided counterpart $K^{straight}$ of K can be defined using high-order nodes $(A_i^{straight})_{d+2\leq i\leq n}$ and the following quantity is considered:

$$\delta = \max_{d+2\leq i\leq n} \frac{||A_i - A_i^{straight}||}{\ell_i},$$

where ℓ_i is the straight edge length of the edge if A_i belongs to an edge, the largest

straight edge length of the face if A_i strictly belongs to a face, or the largest straight edge length of K if A_i is strictly inside K.
The *straightness criterion* is then the following: if δ is lower than ϵ, then the element is considered straight.
In this work, we use $\epsilon = 1\%$.

4.1.1. NASA Rotor 37

This example is a NASA Rotor 37 used for turbomachinery applications. Two different meshes are considered (see Figure 12).

- A coarse mesh (see Figure 12, middle) with an initial number of 2 434 vertices, 13 326 nodes and 10 970 tetrahedra. The initial mesh average quality is 6.16 with a worst quality of 1 682 due to the presence of sharp anisotropic trailing and leading edges on the input rotor surface mesh. Curving the mesh without optimization provides an average quality of 5.3 with a worst quality of 1 729. Optimization in post and pre processing increase average quality to 2.5 and worst quality to 1 346. Additional information about the quality can be found on the top chart in Figure 13. Note that the number of nodes and tetrahedra is changed to respectively 11 598 and 10 862. Also, 1 782 swaps are performed in preprocessing and 1 941 in postprocessing.

- A fine mesh (see Figure 12, bottom) with an initial number of 22 145 vertices, 137 393 nodes and 106 562 tetrahedra. The initial mesh average quality is 2.14 with a worst quality of 111. Note that it is better than with the coarse mesh as the mesh is finer and therefore deals better with trailing and leading edges. Curving the mesh without optimization provides an average quality of 2.15 with a worst quality of 366. Optimization in post and pre processing increase average quality to 1.7 and worst quality to 27.5. Additional information about the quality can be found on the bottom chart in Figure 13. Note that the number of nodes and tetrahedra is changed to respectively 136 924 and 106 093. Also, 9 232 swaps are performed in preprocessing and 9 695 in postprocessing.

In both cases, we clearly observe the benefits of the optimization that improves the average and the worst quality of the final mesh. Note that the worst quality of the final P^2 might be greater than the one of the initial P^1 mesh. This a consequence of the curving process that decreases the quality of straight elements by curving them. We can also observe that the curvature is not propagated a lot in the volume as it is not visible after 2 or 3 layers of elements, see Figure 12, middle and bottom right. This is emphasized in Figure 17 (top) where we color the elements according the *straightness criterion*. Note that in the case of the coarse mesh, 16.01% of the tetrahedra are curved and that in the case of the fine mesh, 7.04% of the tetrahedra are curved. We recall that a tetrahedron is said to be curved if it violates the straightness criterion with a threshold ϵ of 1%. This is an illustration of St Venant's principle which states that the elasticity solution can be divided into transmissive effects and local disturbances.

4.1.2. NASA Common research model aircraft

This example is the NASA Common Research Model (CRM), an aircraft model that is massively used in both experimental and numerical simulations in fluid dynamics (see Figure 14). Again, two meshes are considered:

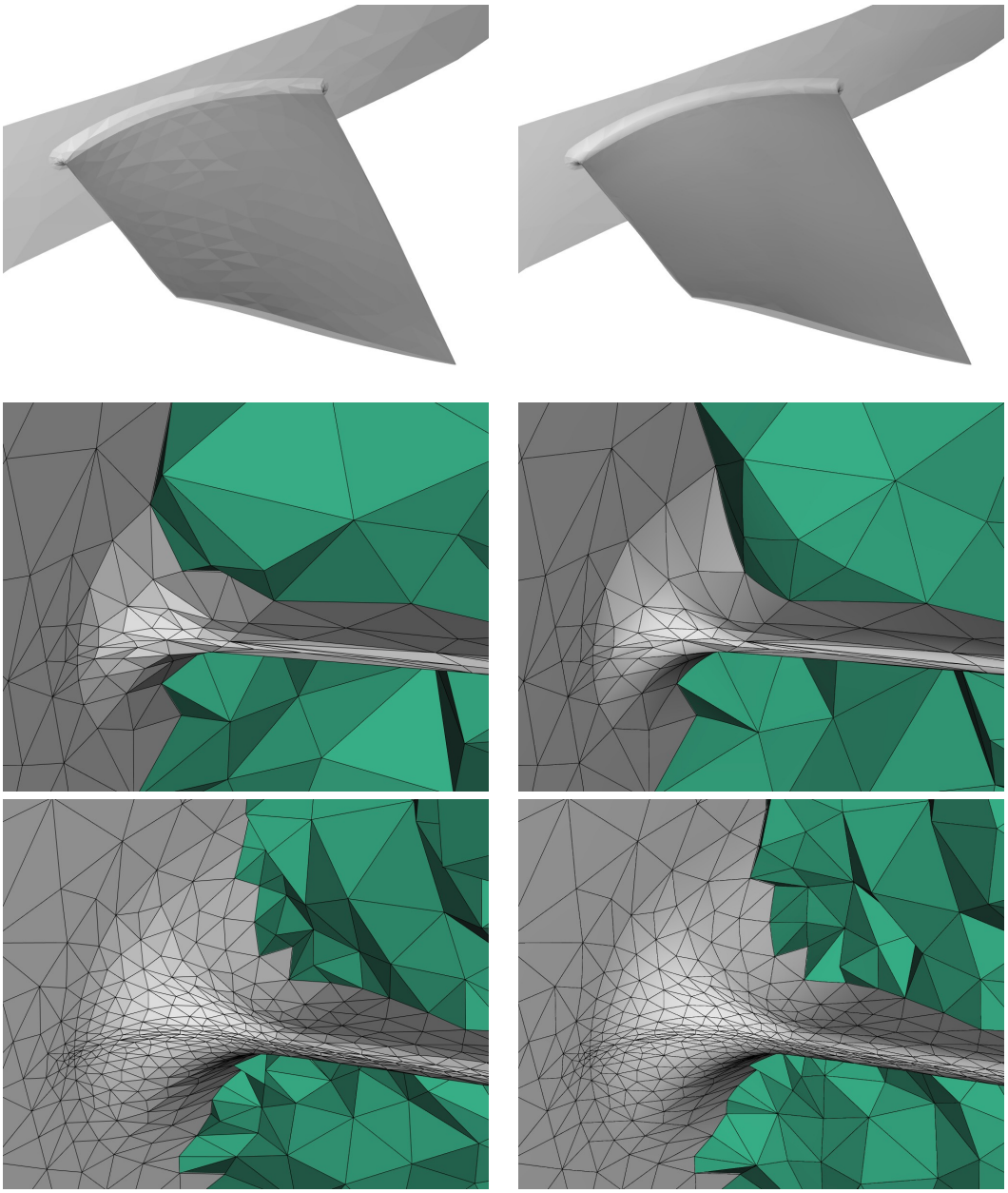

Figure 12: NASA Rotor 37 meshes. From top to bottom, surface, coarse and fine meshes. From left to right, P^1 and P^2 meshes. P^2 meshes are generated with Algorithm 1 using a cubic reconstruction.

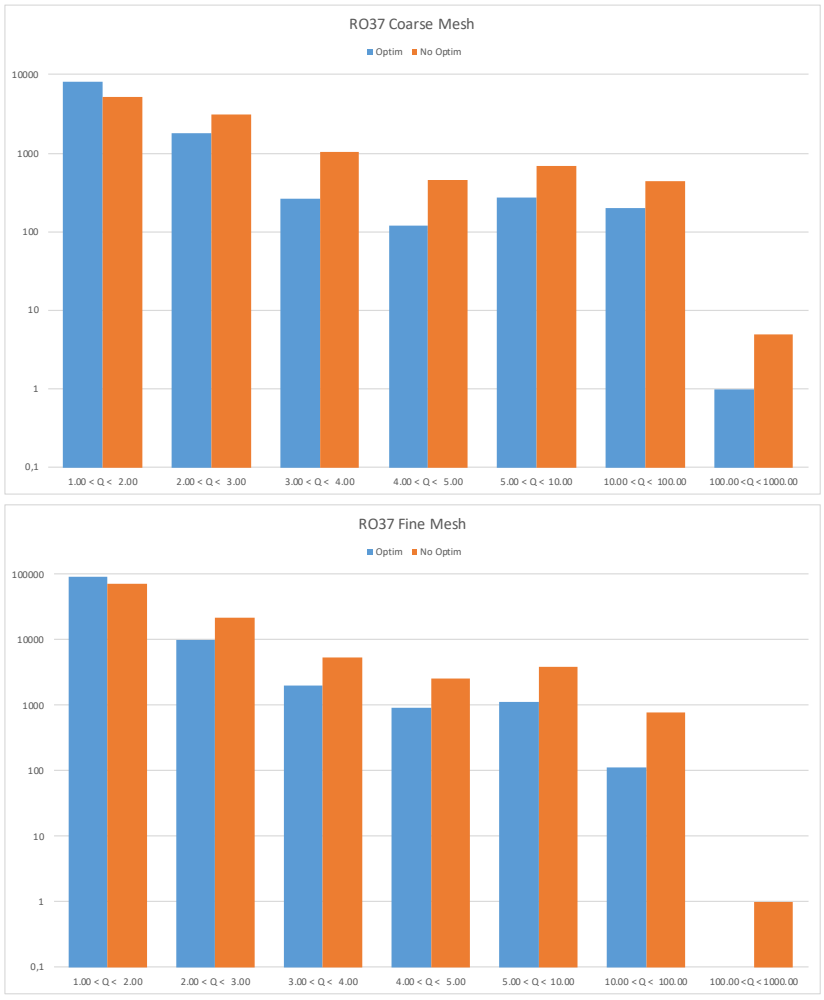

Figure 13: Charts showing the distribution of mesh quality for the two cases with the NASA Rotor 37 with or without optimization. The $y-$axis gives the number of elements. In blue, the charts show the results with optimization. In orange, the charts show the results without optimization.

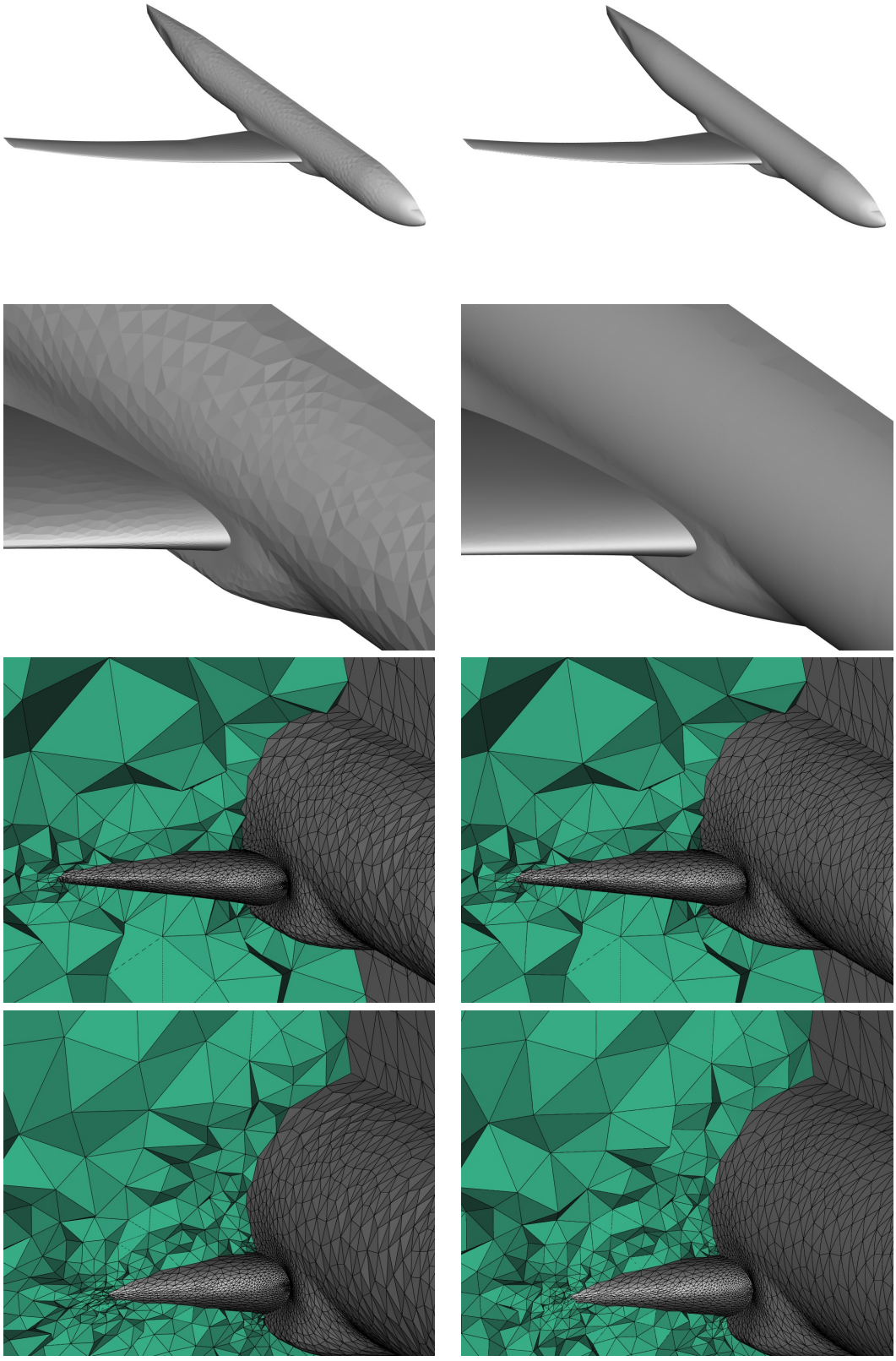

Figure 14: NASA Common Research Model aircraft meshes. From top to bottom, surface, coarse and fine meshes. From left to right, P^1 and P^2 meshes. P^2 meshes are generated with Algorithm 1 using a cubic reconstruction.

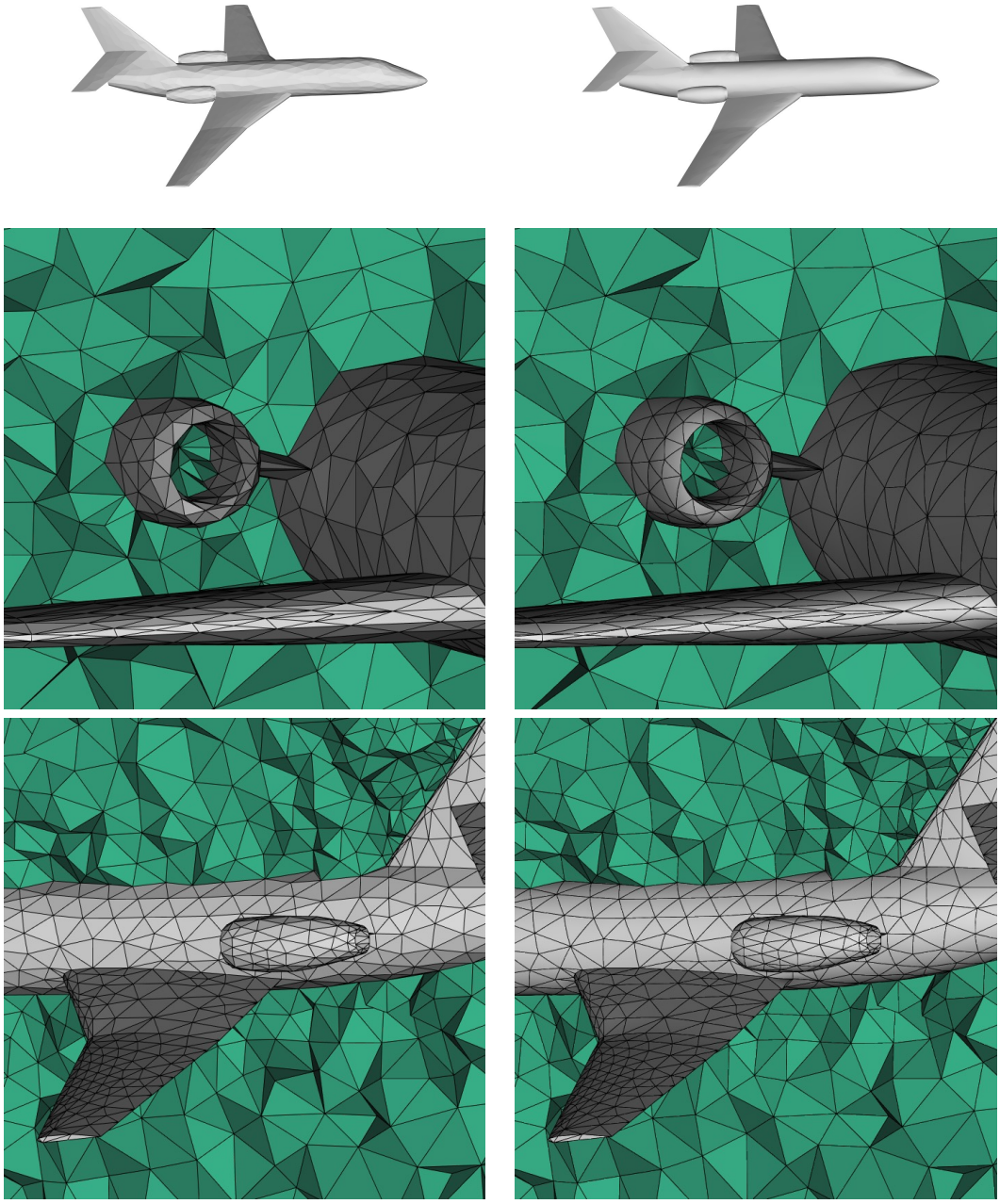

Figure 15: Falcon business jet meshes. From top to bottom, surface and volume meshes. From left to right, P^1 and P^2 meshes. P^2 meshes are generated with Algorithm 1 using a cubic reconstruction.

Figure 16: Charts showing the distribution of mesh quality for the cases involving the NASA Common Research Model and the Falcon with or without optimization. The $y-$axis gives the number of elements. In blue, the charts show the results with optimization. In orange, the charts show the results without optimization.

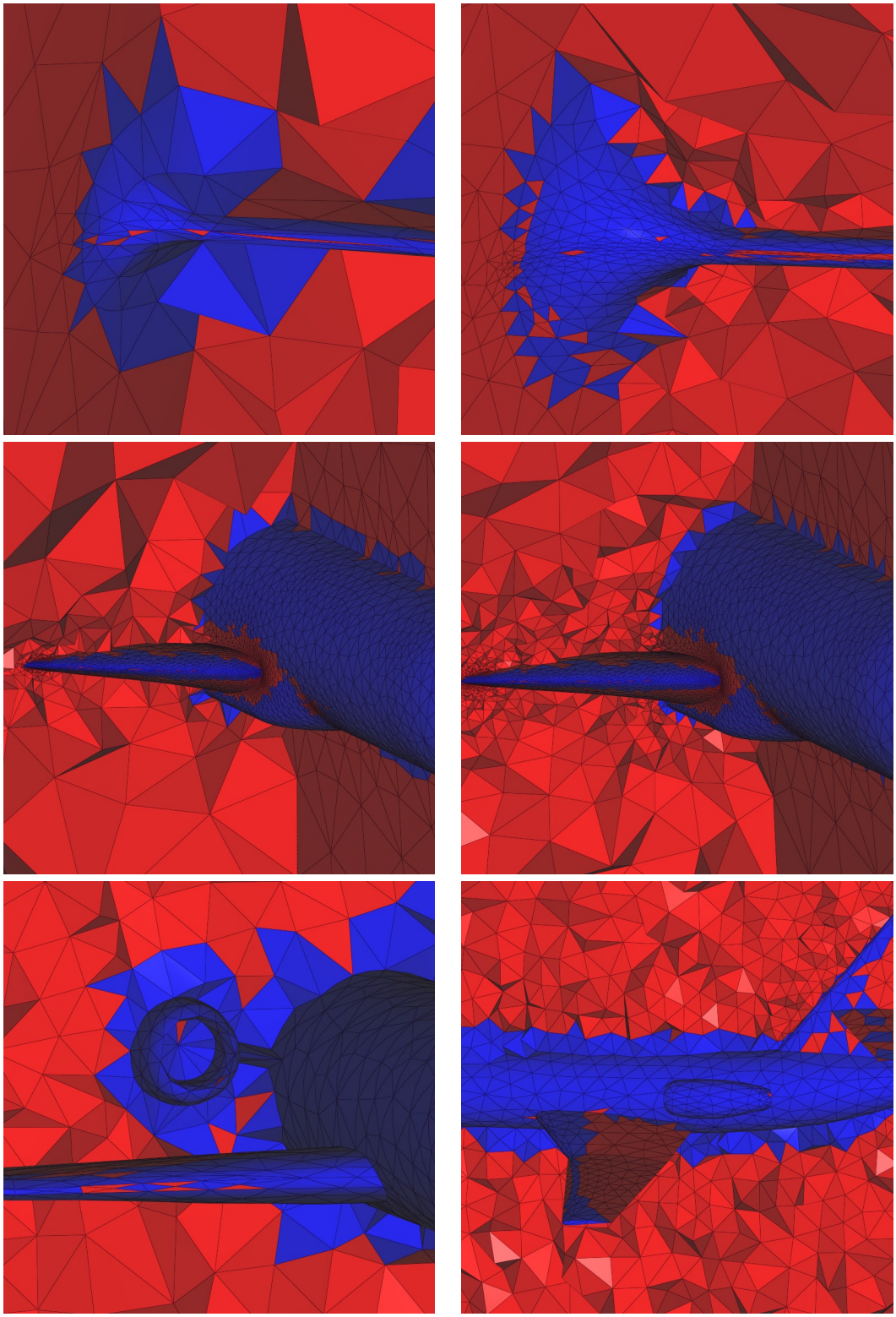

Figure 17: Impact of the mesh curving process for each of the previously studied cases. In red, the volume elements are straight according the straightness criterion of Section 4.1 used with $\epsilon = 1\%$. In blue are shown the volume elements that are curved. On each surface triangle is shown the straightness of the volume element generated from it.

- A coarse mesh (see Figure 14, line 3) with an initial number of 32 479 vertices, 173 468 nodes and 118 012 tetrahedra. The initial mesh average quality is 2.49 with a worst quality of 271 due to the presence of sharp anisotropic trailing and leading edges on the input wing surface mesh. Curving the mesh without optimization provides an invalid configuration with 10 invalid elements. Optimization in post and pre processing increase average quality to 2.13 and worst quality to 271. Additional information about the quality can be found on the top chart in Figure 16. Note that the number of nodes and tetrahedra is changed to respectively 173 744 and 118 288. Also, 10 318 swaps are performed in preprocessing and 10 727 in postprocessing.

- A fine mesh (see Figure 14, line 4) with an initial number of 101 422 vertices, 660 071 nodes and 535 672 tetrahedra. The initial mesh average quality is 2.28 with a worst quality of 1 314. Curving the mesh without optimization provides an invalid configuration with 4 invalid elements. Optimization in post and pre processing increase average quality to 1.4 and worst quality to 1 777. Additional information about the quality can be found on the middle chart in Figure 16. Note that the number of nodes and tetrahedra is changed to respectively 659 545 and 535 146. Also, 44 207 swaps are performed in preprocessing and 46 753 in postprocessing.

In both cases, optimization ensures a valid curved mesh at the end that would not have been obtained without it. In the same way as in the previous example, the curvature is not propagated a lot in the volume as it is not visible after 2 or 3 layers (see Figure 14, line 3 and 4 right and Figure 17, middle where we color the elements according the *straightness criterion*). Note that in the case of the coarse mesh, 15.93% of the tetrahedra are curved and that in the case of the fine mesh, 6.45% of the tetrahedra are curved.

4.1.3. Dassault Falcon aircraft

This example is a notional Dassault Falcon aircraft, a personal business jet that is used by private persons, companies and governments (see Figure 15). For this example we consider a mesh (see Figure 15, line 2) with an initial number of 89 078 vertices, 599 539 nodes and 498 181 tetrahedra. The initial mesh average quality is 1.44 with a worst quality of 21. Optimization in post and pre processing increase average quality to 1.42 and worst quality to 30.66. Additional information about the quality can be found on the bottom chart in Figure 16. Note that the number of nodes and tetrahedra is changed to respectively 599 287 and 497 929. Also, 3 887 swaps are performed in preprocessing and 3 942 in postprocessing. In the same way as in the previous examples, the curvature is not propagated a lot in the volume as it is not visible after 2 or 3 layers (see Figure 15, line 2 and 3 right and Figure 17 bottom, where we color the elements according the *straightness criterion*). Note that in this case, 1.5% of the tetrahedra are curved.

4.2. A connectivity-change moving mesh technique for P^2 elements

In this section, a connectivity-change moving-mesh method inspired from [15] for P^2 meshes is presented. In this case, the initial mesh is a P^2-mesh whose boundary has an initial displacement. Using a linear elasticity analogy, the resolution of the elasticity equation with high-order finite elements gives us a displacement for all the vertices and nodes in the volume. Then the mesh is moved to the new position. The motion of the

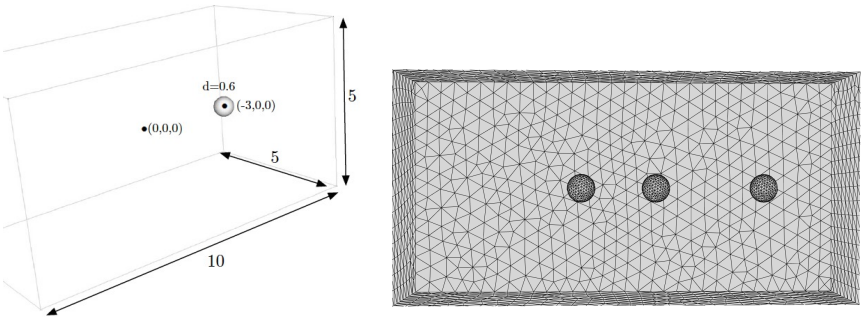

Figure 18: Case of the moving sphere inside a P^2 mesh. Left, description of the geometrical domain. Right, illustration of the three considered positions of the moving sphere where the mesh is analyzed. The sphere is moving from right to left.

vertices can be also enhanced by using a local stiffness factor technique [15]. This technique locally multiplies the tensor σ of linear elasticity equation by a factor proportional to $\mathcal{J}_K(\mathbf{x})^{-\chi}$. χ determines the degree by which smaller elements are rendered stiffer than larger ones. We use $\chi = 1$. Afterwards, connectivity changes are performed on the mesh to improve the *quality* of the elements. It is an efficient way to get rid of any shearing that occurs in the mesh. The high-order moving-mesh algorithm is summarized in Algorithm 2.

Algorithm 2 P^2 moving mesh algorithm

While $(t < T^{end})$

1. Compute body displacement from body translation and rotation data for $[t, t + \Delta t]$ to get a boundary displacement vector $\mathbf{d}_{|\partial\Omega}$ and set it as Dirichlet boundary conditions for the linear elasticity equation.
2. Given the boundary conditions, solve linear elasticity equation on the P^2 mesh with the FEM and deduce a volume displacement vector $\mathbf{d}_{|\Omega}$.
3. If predicted mesh motion is invalid then $\Delta t = \Delta t/2$ and goto 1.
4. Perform P^2 mesh optimization
5. Move the mesh
6. $t = t + \Delta t$

Moving sphere. The first studied case is a moving sphere of radius 0.6 inside a large control volume (see Figure 18, left). At each iteration, the sphere is displaced by 0.08 in x direction. The initial P^2-mesh (see Figure 19, line 1) has an average quality of 1.27 and a worst quality of 2.5. We analyzed the quality of the mesh at three different positions of the moving sphere (see Figure 18, right) with and without connectivity-change: the initial position, the position after 30 iterations (*i.e.* a displacement of 4 radii) and the position after 50 iterations (*i.e.* a displacement of 6.7 radii).

When no connectivity-change is done after each moving step iteration, shearing appears

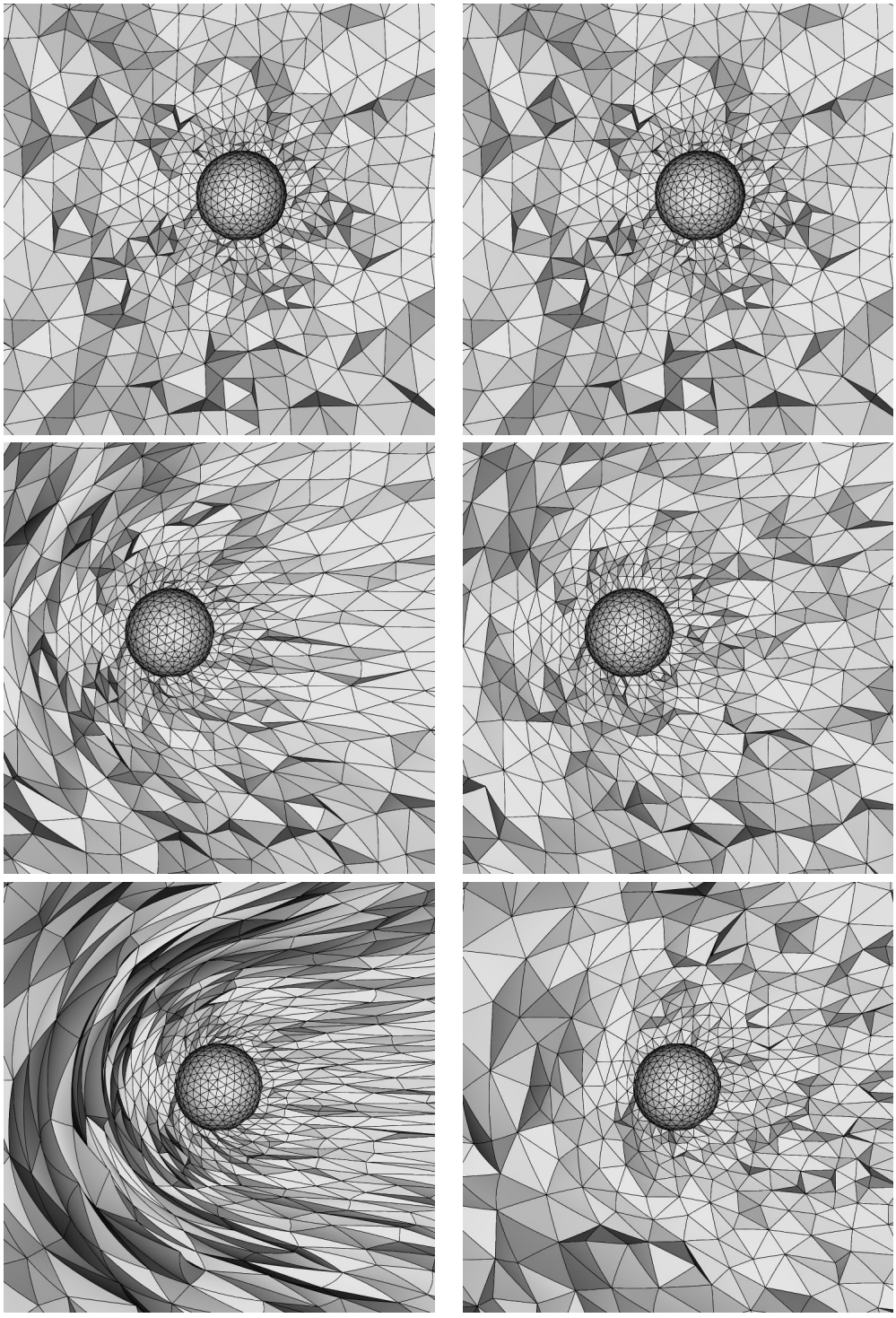

Figure 19: Case of the moving sphere inside a P^2 mesh. From top to bottom, zoom in the vicinity of the sphere at respectively 0 radii, 4 radii and 6.7 radii of displacement. Left, without mesh optimization operators. Right, with P^2 mesh optimization operators.

in the mesh which constrains the displacement. The high-order linear elasticity resolution gives to the elements a curvature that fits to the displacement of the sphere in order to move the object as far as possible when the mesh connectivity is fixed. Indeed, in front of the sphere, the deformed elements fit the shape of the sphere, whereas in the wake, the curvature of elements is made so that the shearing is reduced. This is a good point but the mesh quality decreases drastically (see Figure 19, line 2 and 3 left): after 30 iterations, average quality is 1.64 and worst quality is 5.7 and after 50 iterations, 84 elements are invalid. On the contrary, when P^2 mesh quality-based optimization operators are considered with the moving mesh algorithm at each moving step iteration, the average and the worst quality do not change a lot (see Figure 19, line 2 and 3 right): after 30 iterations, average quality is 1.43 and worst quality is 3.1 and after 50 iterations, average quality is 1.48 and worst quality is 3.5. To get an idea of what happens at every iteration, the evolution of the worst quality of the mesh all along the moving mesh process is shown in Figure 21.

Moving Falcon. The second studied case is a moving falcon aircraft (see figure 15 for the geometry) of size whose bounding box is $x : [0\ 18]\ y : [-9\ 9]\ z : [-1\ 4.75]$. It is also inside a large control volume ($x : [-129.6\ 30]\ y : [-50\ 50]\ z : [-50\ 50]$). The imposed motion is both a translation and a rotation. More precisely, this is a translation of -1.5 in x coupled with a rotation of 1.8 degrees around Ox axis that is done at each iteration. The initial mesh has an average quality of 1.42 and a worst quality of 30.58. Again, three positions are considered to analyze the quality of the P^2-mesh: the initial position, the position after 25 iterations and the position after 50 iterations.

The observations are quite similar but the importance of mesh optimization is more striking. Indeed, due to of the presence of the rotation, the mesh becomes quickly invalid (see Figure 21). Otherwise, we can still say that the use of the linear elasticity allows the motion of the mesh to be good in the vicinity of the falcon, as it can be seen on the line 2 and 3 of Figure 20, on the left, where the mesh is still valid even if no optimization is used. However, the absence of both smoothing and swapping causes a lot of shearing in the front and in the wake of the aircraft. Also, the use of P^2 mesh optimization allows the worst and average quality to remain relatively constant. After 25 iterations (line 2, right of Figure 20), the average quality is of 1.48 and the worst quality is of 26.57 and after 50 iterations (line 3, right of Figure 20) the average quality is of 1.51 and the worst quality is of 32.97. Finally, the global evolution of the quality during this process is shown in Figure 21. It can be noted that the value of the worst quality oscillates in the end. This a consequence of the stretching induced by the rotation of the aircraft. It creates bad quality elements whose optimization takes several iterations to remove.

Ejection door. The initial Algorithm 2 can be improved using the same idea as in [15], that is to say to compute a speed and an acceleration and therefore deduce a quadratic trajectory for all the degrees of freedom. This notably reduces the number of linear elasticity resolution, as the trajectory is performed by interlacing extrapolations based on speed and acceleration and linear elasticity resolutions. The process is summarized in Algorithm 3.

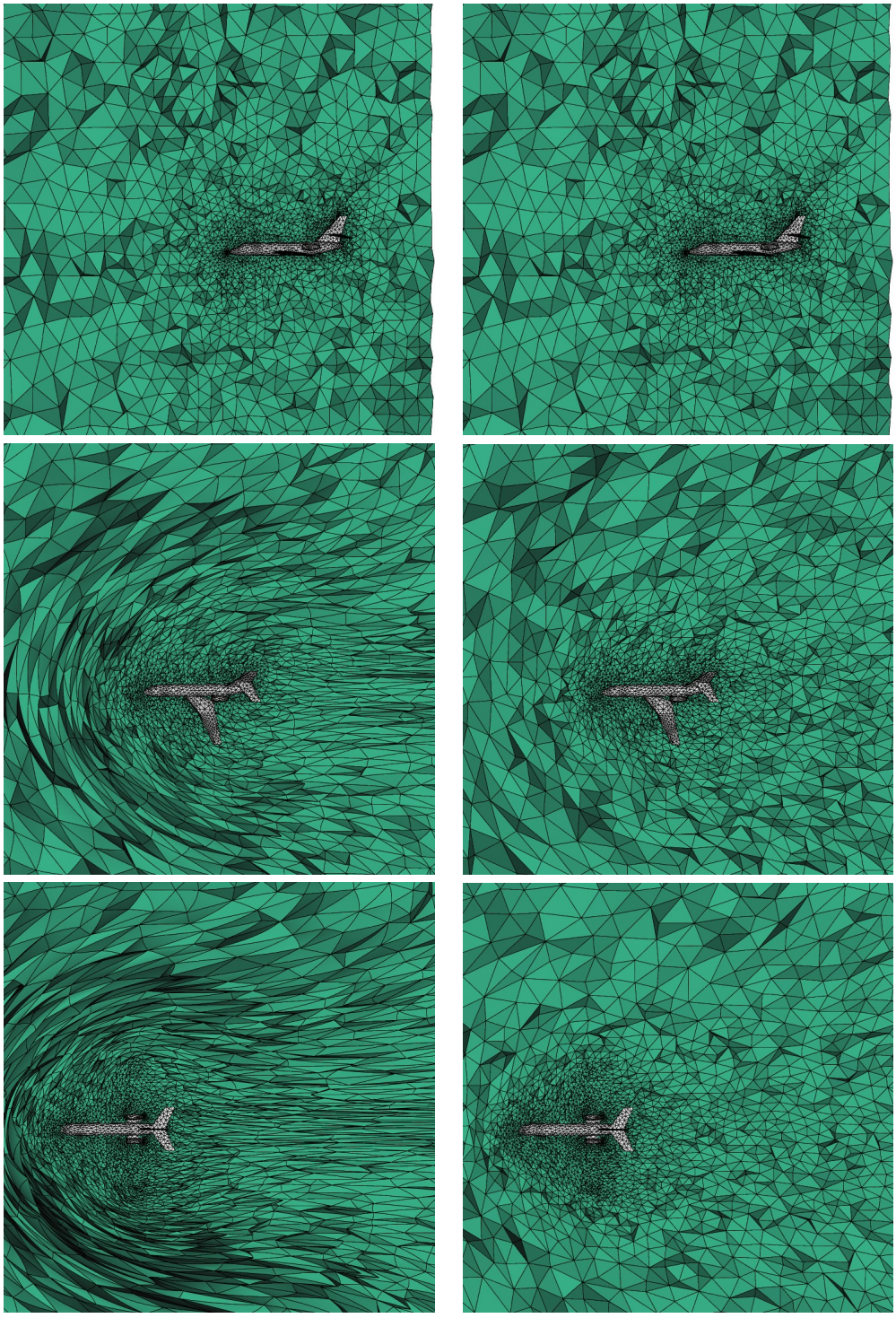

Figure 20: Case of the translating-rotating falcon aircraft inside a P^2 mesh. From top to bottom, zoom in the vicinity of the falcon jet after 0, 25 and 50 iterations of moving-mesh process. Left, without mesh optimization operators. Right, with P^2 mesh optimization operators.

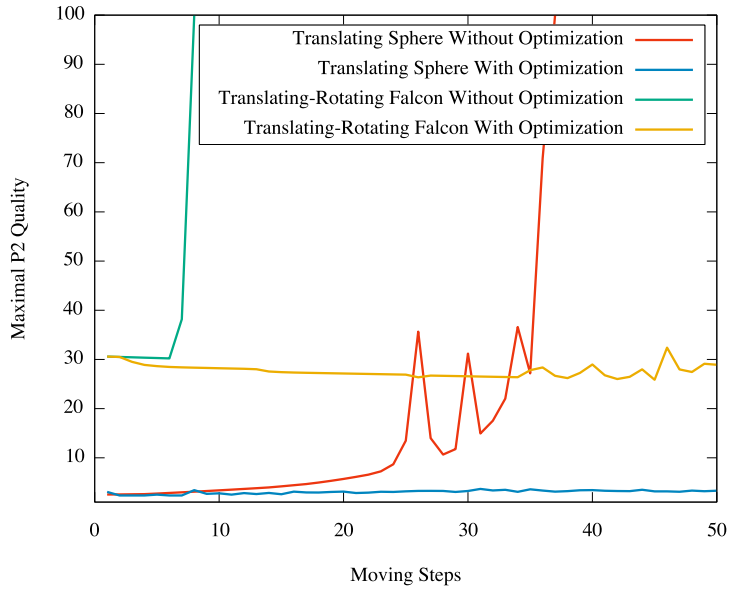

Figure 21: Evolution of the worst P^2 quality of the mesh of both moving sphere and falcon all along the moving mesh process with and without optimization.

Algorithm 3 Improved P^2 Moving Mesh Algorithm with Curved Trajectories

While $(t < T^{end})$

1. Compute body displacement from current body translation, rotation, speed and acceleration data for $[t, t + \Delta t/2]$ and then $[t, t + \Delta t]$. Solve elasticity and obtain $\mathbf{d}(t + \Delta t/2)$ and $\mathbf{d}(t + \Delta t)$
2. Deduce inner vertex speed and acceleration for both displacements $\mathbf{d}(t + \Delta t)$ and $\mathbf{d}(t + \Delta t/2)$
3. If predicted mesh motion is invalid then $\Delta t = \Delta t/2$ and goto 1.
   Else $T^{els} = t + \Delta t$
4. While $(t < T^{els})$
   (a) Get moving mesh step δt thanks to CFL^{geom} and speed data.
   (b) Perform P^2 mesh optimization
   (c) Move the mesh and update vertex speed and acceleration
   (d) $t = t + \delta t$

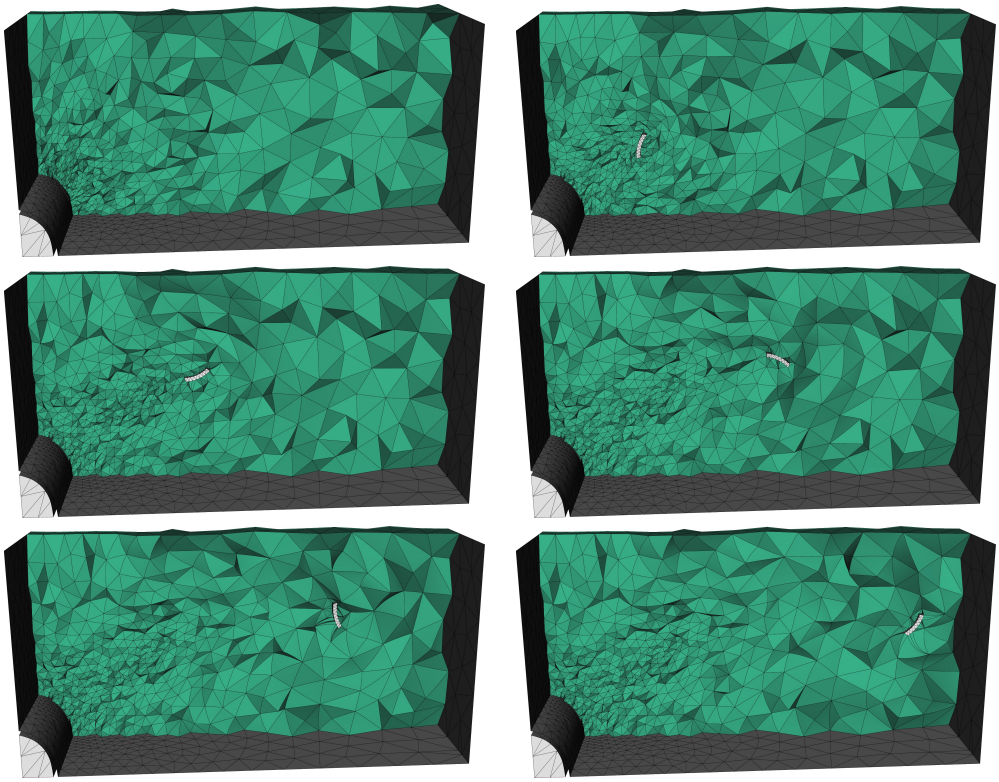

Figure 22: Ejected cabin door test case. Snapshots of the displacement at time 0, 0.32, 0.64, 0.96, 1.28 and 1.6 from left to right and top to bottom

This improved algorithm is used to perform a more complex test case than the two previous ones. The problem is the door ejection of an over-pressurized aircraft cabin. This is a usual industrial benchmark for aircraft designers whose aim is to evaluate when the door hinge will yield under cabin pressure. Indeed, the trend for new aircrafts is to lower the cabin altitude (*e.g.*, higher cabin pressure). The difficulty is that the geometry is anisotropic which forces to deal with low quality elements.

The door movement is shown in Figure 22 and corresponds to a physical time from 0s to 1.6s. For this test case, a total of 811 linear elasticity resolution have been performed and 5 519 moving steps have been done. The worst detected quality all along the process is of 20.097 and a total number of 631 764 swaps have been done. In Figure 23, the evolution of the average and worst quality is shown. It can be seen that the average quality is maintained relatively constant all along the process while the worst quality is slightly decreased all along the process. The latter is due to the initial configuration where badly shaped elements are located between the door and the body of the aircraft.

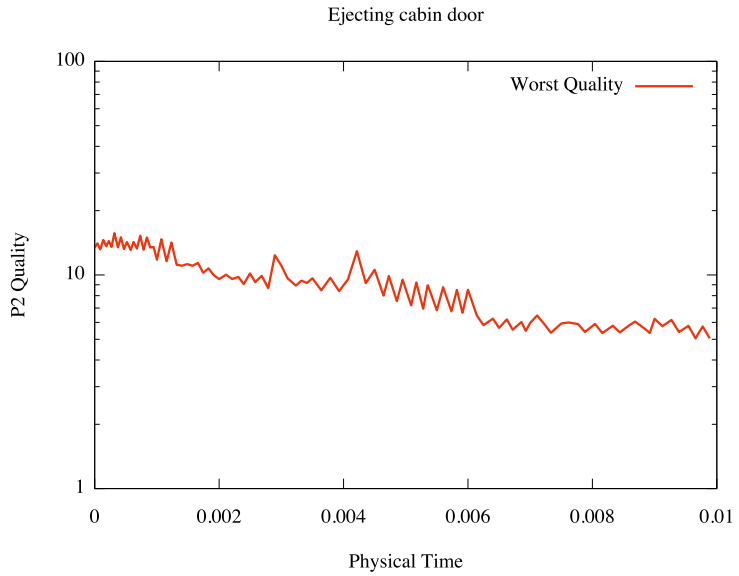

Figure 23: Evolution of the worst P^2 quality of the mesh of the door-ejection case all along the moving mesh process.

5. Conclusion and perspectives

P^2 mesh quality-based optimization operators have been presented. These operators ensure a valid P^2 mesh generation starting from a P^1 mesh and enable to deal with connectivity-change P^2 moving-mesh methods. Note that all these developments are suitable for isotropic meshes and surfaces with small anisotropy but are not designed for full anisotropic volume meshes and do not work as is with boundary layer meshes. The moving-mesh method gives similar results in term of quality as in P^1 which is promising for future research. Isotropic degree two meshes were only the first step, further developments will be to generalize it to any higher-order meshes and to deal with anisotropy using metric fields. When it comes to boundary layer meshes, the goal is to extend the closed-advancing boundary layer mesh generation method of [27] to high-order meshes in order to generate directly curved boundary layer meshes. To this end, it is required to:

- Start from an initial high-order mesh that is obtained using the method of Section 4.1
- Consider the connectivity-change moving mesh method for high-order mesh presented in Section 4.2 with curved trajectories to deform the initial high-order mesh when the boundary layer mesh is inflated inside the domain
- Generate directly high-order elements in the boundary layer when it is inflated using the advancing layer approach presented in [27].

The future work to do is the last item. The advancing layer method will be modified to take into account the high-order boundary layer elements. The new position of the nodes in the boundary layer will be given using the same process as the one for proposing the new position for the vertices. High-order quality functions will be used to check the quality of the boundary layer elements when they are generated.

Note that more accurate normals will be obtained as they will be computed on the high-order mesh instead of a P^1-straight mesh. This is an important point as the quality of the boundary layer is highly dependent on the accuracy of the normal computations [27]. Also, curved boundary layer mesh generation will not need the creation of a new tool and should be more robust and efficient as it directly controls the high-order mesh quality.

Acknowledgment

This work was supported by a public grant as part of the *Investissement d'avenir* project, reference ANR-11-LABX-0056-LMH, LabEx LMH and ANR IMPACTS, reference ANR-18-CE46-0003.

References

[1] P. Ciarlet, The Finite Element Method for Elliptic Problems, North-Holland Publishing Company, 1978.

[2] J. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: algorithms, analysis and applications, Springer Publishing Company, Incorporated, 2008.

[3] J. Vanharen, G. Puigt, X. Vasseur, J.-F. Boussuge, P. Sagaut, Revisiting the spectral analysis for high-order spectral discontinuous methods, Journal of Computational Physics 337 (2017) 379 – 402.

[4] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, Journal of computational physics 138 (2) (1997) 251–285.

[5] S. Dey, R. O'Bara, M. Shephard, Curvilinear mesh generation in 3D, Comput. Aided Geom. Design 33 (2001) 199–209.

[6] S. Sherwin, J. Peiró, Mesh generation in curvilinear domains using high-order elements, International Journal for Numerical Methods in Engineering 53 (1) (2002) 207–223.

[7] M. Fortunato, P.-O. Persson, High-order Unstructured Curved Mesh Generation Using the Winslow Equations, J. Comput. Phys. 307 (2016) 1–14.

[8] D. Moxey, D. Ekelschot, Ü. Keskin, S. Sherwin, J. Peirò, High-order curvilinear meshing using a thermo-elastic analogy, Computer-Aided Design 72 (2016) 130 – 139.

[9] R. Hartmann, T. Leicht, Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration, International Journal for Numerical Methods in Fluids 82 (6) (2016) 316–333.

[10] S. L. Karman, J. T. Erwin, R. S. Glasby, D. Stefanski, High-order mesh curving using WCN mesh optimization, in: 46th AIAA Fluid Dynamics Conference, AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, 2016, p. 3178.

[11] E. Ruiz-Gironès, X. Roca, J. Sarrate, High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation, Computer-Aided Design 72 (2016) 52 – 64, 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[12] T. Toulorge, C. Geuzaine, J.-F. Remacle, J. Lambrechts, Robust untangling of curvilinear meshes, Journal of Computational Physics 254 (2013) 8 – 26.

[13] P. L. George, H. Borouchaki, Construction of tetrahedral meshes of degree two, International Journal for Numerical Methods in Engineering 90 (9) (2012) 1156,1182.

[14] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, Journal of Computational Physics 233 (15) (2013) 359–372.

[15] F. Alauzet, A changing-topology moving mesh technique for large displacements, Engineering with Computers 30 (2) (2014) 175–200.

[16] P. Bézier, Courbes et surfaces, Hermès, 1986.
[17] H. Borouchaki, P. L. George, Meshing, Geometric Modeling and Numerical Simulation 1: Form Functions, Triangulations and Geometric Modeling, John Wiley & Sons, 2017.
[18] P. de Casteljau, Outillages, méthodes, calcul, André Citroën Automobiles SA, Paris.
[19] P. L. George, H. Borouchaki, F. Alauzet, P. Laug, A. Loseille, L. Maréchal, Meshing, Geometric Modeling and Numerical Simulation 2: Metrics, Meshes and Mesh Adaptation, John Wiley & Sons, 2019.
[20] P. J. Frey, P. L. George, Mesh generation: application to finite elements, John Wiley & Sons, 2008.
[21] T. Toulorge, J. Lambrechts, J.-F. Remacle, Optimizing the geometrical accuracy of curvilinear meshes, Journal of Computational Physics 310 (2016) 361 – 380.
[22] N. Barral, Time-accurate anisotropic mesh adaptation for three-dimensional moving mesh problems, Ph.D. Thesis, Université Pierre et Marie Curie - Paris VI (Nov. 2015).
[23] D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical Programming 45 (1) (1989) 503–528.
[24] M. Turner, J. Peirò, D. Moxey, A Variational Framework for High-order Mesh Generation, Procedia Engineering 163 (Supplement C) (2016) 340 – 352, 25th International Meshing Roundtable.
[25] P. L. George, F. Hecht, E. Saltel, Automatic mesh generator with specified boundary, Comput. Meth. Appl. Mech. Engrg.92 (1991) 269–288.
[26] A. Vlachos, P. Jörg, C. Boyd, J. L. Mitchell, Curved PN Triangles, in: Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D '01, ACM, New York, NY, USA, 2001, pp. 159–166.
[27] F. Alauzet, A. Loseille, D. Marcum, On a robust boundary layer mesh generation process, in: 55th AIAA Aerospace Sciences Meeting, AIAA Paper2017-0585, Grapevine, TX, USA, 2017, p. 0585.

Appendix

Derivation of the terms of the quality function

In this appendix, we give some details on the strategy used in [19] to tailor a quality function that behaves as expected for the case of a P^2-triangle. That is to say, a function that is between one and the infinity and that has an optimal value of 1 when the element is regular/equilateral. The starting point of the analysis is the following : in P^1, a commonly used quality function ([20]) is:

$$Q = \alpha \frac{h}{\rho} = \alpha \frac{hp}{S} \in [1, +\infty),$$

where S is the surface of the element, h the largest edge length, p its half perimeter, ρ its inradius and α a normalization factor so that $Q = 1$ for equilateral triangles.
A natural idea is to extend this notion to the P^2-triangle and to compute both surface and lengths on the P^2 element. However, such a function is unable to detect the invalid elements. To overcome this issue, the following dimensionless factor $\frac{J_{max}}{J_{min}}$ known as *scaled Jacobian* is considered. This is the ratio between the largest and smallest values of the Jacobian. As the Jacobian is a polynomial, it is not easy to compute its bounds. A solution is consequently to replace it by $\frac{N_{max}}{N_{min}}$ where N_{max} and N_{min} are the largest and smallest values of the control coefficients of the Jacobian (*i.e.* the coefficients of the Jacobian in the Bernstein basis) and these values bound the Jacobian (*i.e.* $N_{min} \leq J_{min}$ and $N_{max} \geq J_{max}$). Note this ratio equals 1 for straight-sided elements and is able to detect the Jacobian-based distortion of the elements. Now, as we want to limit its impact on the value of the quality function, the ratio is replaced by its square root. This way, it allows more easily slightly curved elements to be elements of good quality. This gives us the following quality function:

$$Q = \alpha \frac{hp}{S} \sqrt{\frac{N_{max}}{N_{min}}}.$$

In P^2, S can be computed using Formula (4), but h and p cannot be exactly computed. They are consequently replaced by approximated values that are based on the length of a linear decomposition (using control points) of the P^2 edges.

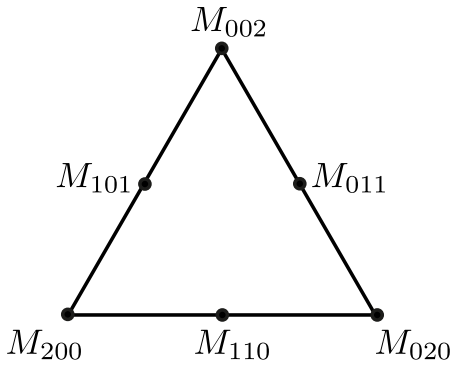

Figure 24: P^2-triangle used for the study of the quality function.

Let us analyze this function in more details by considering the equilateral triangle defined in Figure 24 with its P^2-nodes. Three tests are made:

- The first one consists in moving the node M_{110} on the straight edge $[M_{200}, M_{020}]$. As expected, the quality function equals one only when M_{011} is in the middle of the edge, is greater than one elsewhere and is symmetrical (it has the same behavior if it goes to M_{200} or M_{020}).
- The second one consists in moving the node M_{110} in the orthogonal direction of the straight edge $[M_{200}, M_{020}]$. Going by 10% of the edge length inside the triangle gives a quality value of 1.112 and going by 10% of the edge length outside the triangle gives a quality value of 1.416.
- The third one consists in slightly curving the edge containing M_{110} of 10% inside and considering displacements of M_{101}. By doing so, it is possible to find values of the quality functions that are lower than 1.

From these simple analysis, we note that the quality function is not contained in $[1, \infty)$. For this reason, a third term is added:

$$Q = \alpha \frac{hp}{S} \frac{\max(S_{P^1}, S)}{\min(S_{P^1}, S)} \sqrt{\frac{N_{max}}{N_{min}}},$$

where S_{P^1} is the area of the triangle defined with the vertices (but not the nodes). With this third term, the quality function behaves as expected and gives values only in $[1, \infty)$. A more complete analysis of the behavior is done in [19].
The generalization to higher degrees is straightforward and the extension to tetrahedra is possible if we replace S by the volume and p by the external surface of the linear decomposition of the high-order element (for a tetrahedron in P^1, $\rho = \frac{V}{S}$).
This function has been tested on several high-order simplicial meshes in 2D and 3D and

has been proven to be a good criteria for generating high quality meshes. A generalization of this quality function to high-order hybrid meshes has also been successfully implemented.